AFRL-HE-WP-TR-2006-0014

STINFO COPY

AIR FORCE RESEARCH LABORATORY

Hybrid Simulation/Analytic Models for Military Supply Chain Performance Analysis

Stephen Farris
Manuel D. Rossetti

University of Arkansas
Department of Industrial Engineering
4207 Bell Engineering Center
Fayetteville, AR 72701

October 2004

Interim Report for November 2002 to October 2004

20060418091

Approved for public release; distribution is unlimited.

Human Effectiveness Directorate
Warfighter Readiness Research Division
Logistics Readiness Branch
2698 G Street
Wright-Patterson AFB OH 45433-7604

NOTICES

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data, does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them

This report was cleared for public release by the Air Force Research Laboratory Wright Site Public Affairs Office (AFRL/WS) and is releasable to the National Technical Information Service (NTIS). It will be available to the general public, including foreign nationals.

Please do not request copies of this report from the Air Force Research Laboratory. Additional copies may be purchased from:

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161

Federal Government agencies and their contractors registered with the Defense Technical Information Center should direct requests for copies of this report to:

Defense Technical Information Center
8725 John J. Kingman Road, Suite 0944
Ft. Belvoir, VA 22060-6218

DISCLAIMER

This Technical Report is published as received and has not been edited by the Air Force Research Laboratory, Human Effectiveness Directorate.

TECHNICAL REVIEW AND APPROVAL

AFRL-HE-WP-TR-2006-0014

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER

//SIGNED//

DANIEL R. WALKER, Colonel, USAF
Chief, Warfighter Readiness Research Division
Human Effectiveness Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
October 2004	Interim	November 2002 - October 2004

4. TITLE AND SUBTITLE
Hybrid Simulation/Analytic Models for Military Supply Chain Performance Analysis

5a. CONTRACT NUMBER
F33615-99-D-6001 DO#23

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
63231F

6. AUTHOR(S)
Stephen Farris, Manuel D. Rossetti

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
49230031

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Arkansas
Dept. of Industrial Engineering
4207 Bell Engineering Center
Fayetteville, AR 72701

8. PERFORMING ORGANIZATION REPORT NUMBER
BSIT 0201

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory, Human Effectiveness Directorate
Warfighter Readiness Research Division
Air Force Materiel Command
Logistics Readiness Branch
Wright-Patterson AFB OH 45433-7604

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-HE-WP-TR-2006-0014

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.
AFRL/WS-06-0405, 14 Feb 06

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This research is intended to extend the knowledge base concerning logistical network modeling and design. Basic research techniques were developed to begin to model logistical networks within a hybrid simulation/analytic framework. The first step in this process is to develop robust approximations for portions of large-scale simulation models. This research examined the novel idea of utilizing neural networks as a meta-modeling technique to replace specific aspects of a simulation. This work started with the replacement of queueing stations within a logistics network. Any logistics network can be formulated as a network of material flowing through processes requiring resources. A new methodology was developed for forming approximations and improved approximators for queueing stations within a logistics network. The motivation for developing this approximation is the integration of such approximation in hybrid simulation/analytic methods for evaluating logistic networks. Future work should investigate the performance of these approximations within the larger logistical network context.

15. SUBJECT TERMS
Logistical Network Modeling, Logistical Network Design, Hybrid Simulation/Analytic Framework

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Cheryl L. Batchelor
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR	82	19b. TELEPHONE NUMBER *(include area code)* 937-656-4392

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. 239.18

THIS PAGE LEFT INTENTIONALLY BLANK

Contents

Figures

Tables

Exhibits

Executive Summary

This research is intended to extend the knowledge base concerning logistical network modeling and design. Basic research techniques were developed to begin to model logistical networks within a hybrid simulation/analytic framework. The first step in this process is to develop robust approximations for portions of large-scale simulation models. In this research, we examine the novel ideal of utilizing neural networks as a meta-modeling technique to replace specific aspects of a simulation. In this work, we start with the replacement of queueing stations within a logistics network.. Any logistics network can be formulated as a network of material flowing through processes requiring resources. We develop a new methodology for forming approximations and develop improved approximators for queueing stations within a logistics network.

We demonstrate the use of neural networks to close the gap between the output of Whitt's GI/G/m approximation and the results obtained via simulations of a GI/G/m queue. Once the neural network has been trained, we can feed in the parameters and resulting output of Whitt's GI/G/m approximation, along with additional information describing the arrival and service distributions of the queue, and yield an acceptably accurate approximation for the expected wait time in a GI/G/m queue. These approximations can be embedded in parametric decomposition algorithms for the logistics network as a whole. The motivation for developing this approximation is the integration of such approximation in hybrid simulation/analytic methods for evaluating logistic networks. The neural network approximation developed easily beats Whitt's approximation for correlated arrivals, but further investigation is needed to assure a robust approximation. Future work should investigate the performance of these approximations within the larger logistical network context.

1 Introduction

The purpose of this document is to discuss the findings associated with the project entitled, "Hybrid Simulation/Analytic Models for Military Supply Chain Performance Analysis." This project represents a joint effort between the University of Arkansas Center for Engineering Logistics and Distribution (CELDi) and the Air Force Research Laboratory (AFRL)'s Human Effectiveness Directorate. The intended audience for this report is CELDi researchers and AFRL personnel.

The use of information technology will allow for total asset visibility and the integration of logistical operations and logistical planning so mission execution will be based upon anticipatory and just-in-time strategies. Logistical planners must have the ability to evaluate rapidly the performance of various deployment scenarios in order to optimize the delivery of the correct materials at the correct times in the correct quantities to the correct locations. This research is intended to provide basic research concerning logistical network modeling and design. Techniques are examined to model logistical networks within a hybrid simulation/analytic framework. In this research, the simulation/analytic framework is examined at the subcomponent level. In simulation/analytic approaches, the network is first decomposed into subcomponents to which the most appropriate modeling technique is applied.

The long-term goal of this basic research is to examine the feasibility of using hybrid simulation/analytic techniques within logistical performance analysis in order to speed up the execution of logistical planning scenarios while maintaining the integrity of the performance predictions. We define the integrity of the performance prediction as consisting of the accuracy and the precision of the resulting predictions. These methods may affect the accuracy of the predictions by introducing bias. The bias may be the result of decreased model validity; that is, the models may lose some of their representation power, or the bias may be due to statistical bias introduced by approximation error. Because these methods incorporate simulation, the precision of the predictions may also be affected. Any loss of precision may be the result of the effect of statistical sampling error inherent in any simulation-based method.

The overall context of the problem is a supply system consisting of operational (production), transportation and storage components. The operational components of the system perform operations on units of work that flow through the system. Operational components involve the use of constraining resources that add value in some manner to the final end-task to be accomplished by the system. Transportation components are primarily involved in the movement of work within the system. Storage components are used to hold intermediate work products to gain efficiency in the use of the timing of operational or transportation resources. These types of systems are generally regarded as queueing systems or more generally as stochastic processing networks.

For example, a multi-indenture, multi-echelon spare parts inventory system falls into the general class of systems considered within this research. A weapon system such as an aircraft may consist of many different parts. The aircraft are subject to multiple processing steps (operations) as they are made ready to fly missions. In addition, the parts move through repair facilities organized into repair operations while transportation elements assist in moving the parts both intra-facility and inter-facility. *Figure 1.1* illustrates a simple spare parts supply chain that contains these elements.

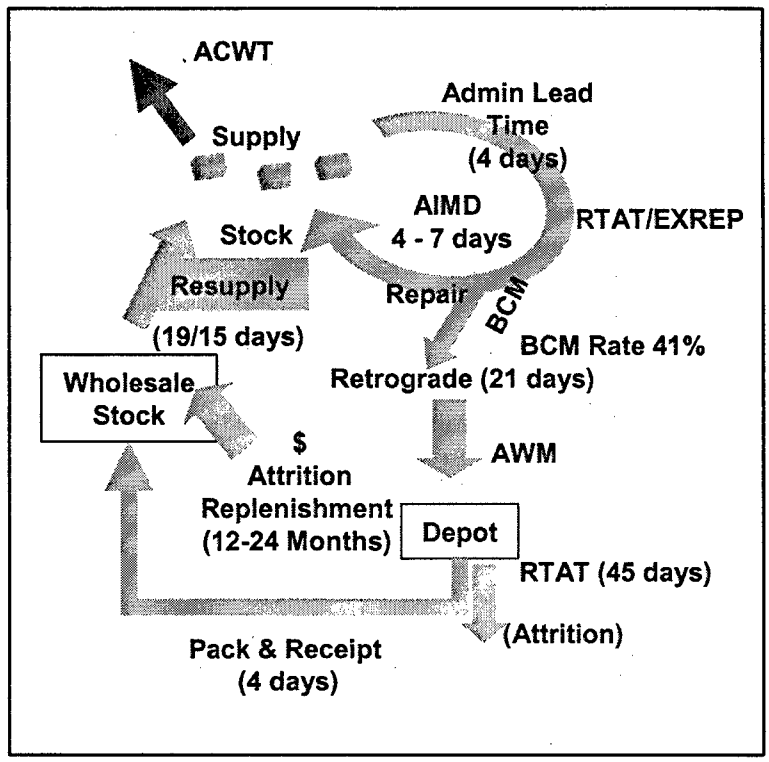

Figure 1.1: Representative Spare Parts Supply Chain

Mathematical models of queueing and inventory systems, and more generally of stochastic processing networks, are useful in analyzing and predicting logistic system performance. Two primary challenges facing the application of queueing and stochastic inventory theory approaches to modeling complex logistical systems are: 1) the inability to adequately represent complex routing and scheduling, and 2) detailed mathematical representations causing difficult-to-compute theoretical results. When easy-to-compute closed-form analytical results are unavailable, two approaches have been used to analyze system behavior: discrete event simulation and approximations. Simulation can be costly, time-consuming, and produce results that depend upon statistical sampling techniques. In addition, computation times for adequately analyzing large-scale logistical networks can also be computationally infeasible. Queueing and inventory model approximations are mathematical models that approximate the behavior of general classes of models for which analytical solutions are unknown or very difficult to compute. Advances in network approximations, especially in parametric decomposition approaches, have made for increased use of such techniques in modeling supply chains.

As an alternative to queueing network models, simulation can provide the level of detail required to capture the complex routing and scheduling necessary for modeling the resource contention involved in such networks. While simulation allows modeling flexibility, it creates difficult-to-overcome data requirements and can be extremely computationally burdensome, especially when the simulation must be executed repeatedly in a planning/optimization exercise.

Traditional queueing and inventory analytic approaches to logistical network design must be made more amendable to the requirements of real logistical planners. Basic research in queueing and inventory approximations and combined simulation/analytic techniques may provide the necessary models to predict the behavior of complex logistical networks and serve as a basis for the development of logistical software planning and optimization tools. This research examines the major issues related to combining simulation and analytic techniques at the subcomponent level. For example, we examine the following issues:

- What types of single-station approximations are the most appropriate?
- How can better (more robust) single-station approximations be developed?
- How can the single-station approximations be combined to model logistical networks?
- How can the analytical methods be properly combined with simulation techniques in the most appropriate manner?
- What is the modeling integrity provided by these techniques?

The rest of this document presents the results of our investigation into these issues. We begin with a review of some relevant literature and then present our results related to the single-station models. A discussion of network modeling approaches is then presented. Finally, we wrap up with our results concerning the integrity of these approaches and conclude with some areas for future research.

2 Literature Review

The purpose of this section is to provide an overview of the literature used during this project.

2.1 Analytical Methods

As previously mentioned, the analysis of stochastic processing networks begins with an ability to predict the performance of a (single-station) queue. These queueing systems can be configured into a network of queues to represent the system. In order to predict the system's performance, several input parameters are needed. Whitt (1983) described the Queueing Network Analyzer (QNA) software and used c_a^2, c_s^2, and ρ to predict the performance of the queue, where c_a^2 is the squared coefficient of the arrival rate, c_s^2 is the squared coefficient of variation of the service time, and ρ is the traffic intensity. Other required inputs to the model are the number of servers (m) and the arrival rates and service rates, which are denoted as λ and μ, respectively.

Springer and Makens (1991) characterize finite queues through the use of k, which is the system buffer capacity. Since Whitt's (1983) model assumes no capacity constraints, the parameter k does not exist for QNA. A slight variation of this procedure is presented in Whitt (1994), in which variability functions, instead of variability parameters, are used as inputs to the analytical models.

2.1.1 Whitt's 1983 QNA Model

Whitt (1983) presented the Queueing Network Analyzer, or QNA, software. The purpose of QNA is to approximate congestion measures for a network of queues. The software is intended for use on non-Markov networks, so arrival processes do not have to be Poisson and service times do not have to be exponential. He assumes each node in the network is stochastically independent. The arrival process to each node is partially characterized by a few parameters, which are represented by linear equations. The system of linear equations is solved to determine the internal flow parameters. Once these parameters have been defined and a routing matrix has been provided, each node is analyzed separately. Calculus transforms the parameters that define each queue and node to represent operations such as merging, splitting and departure.

Whitt (1983) presents calculations for network throughput (λ), the departure rate from the network (d), the total rate of service completions (s), and the overall congestion of the network (EN).

The throughput is equivalent to the total external arrival rate, λ_0.

$$\lambda_0 = \lambda_{01} + \ldots + \lambda_{0n}$$

The departure rate from the network is given by the following equation:

$$d = \sum_{i=1}^{n} d_i = \sum_{i=1}^{n} \lambda_i \gamma_i \left(1 - \sum_{j=1}^{n} q_{ij}\right)$$

where λ is the total arrival rate to node j,
γ is the multiplicative factor of customer creation at the node, and
q is the routing matrix.

The total rate of service completions, s, is:

$$s = \sum_{i=1}^{n} s_i = \sum_{i=1}^{n} \lambda_i \gamma_i$$

The overall congestion of the network is:

$$EN = EN_1 + \ldots + EN_n$$

The results of the performance evaluation of the Queueing Network Analyzer can be found in Whitt (1983).

2.1.2 Whitt's 1993 Model

Whitt's (1993) approximations are presented for the GI/G/m queue. The model presented has unlimited waiting room, m identical parallel servers, and the queue operates on a first-come first-served basis. All service and inter-arrival times are independent, identically distributed random variables with general distributions specified by their first two moments. The arrival process is

specified by λ and c_a^2, which are the arrival rate and the squared coefficient of variation of inter-arrival time. The service process is specified by τ and c_s^2, which are the service rate and the squared coefficient of variation of service time. Expanding on Whitt's earlier QNA model, service times may be non-exponential, and arrival processes may be non-Poisson. Approximations are made for the non-Markovian routing network through the process of parametric decomposition. Other assumptions made by Whitt (1993) are that traffic intensity, ρ, is less than 1, and that service rate, τ, is equal to 1. Traffic intensity, ρ, is equal to $\lambda\tau/m$.

Whitt (1993) focuses on approximations for the expected waiting time of a customer before beginning service, or *EW*. The formula he presents on page 125 of Whitt (1993) for *EW* is:

$$EW\left(\rho, c_a^2, \mathrm{c}_\mathrm{s}^2, \mathrm{m}\right) \approx \phi\left(\rho, c_a^2, c_s^2, m\right)\left(\frac{c_a^2 + c_s^2}{2}\right) EW(M/M/m)$$

where $\phi\left(\rho, c_a^2, c_s^2, m\right) =$

$$\begin{cases} \left(\dfrac{4\left(c_a^2 - c_s^2\right)}{4c_a^2 - 3c_s^2}\right)\phi_1(m,\rho) + \left(\dfrac{c_s^2}{4c_a^2 - 3c_s^2}\right)\psi\left(\left(c_a^2 + c_s^2\right)/2, m, \rho\right), & c_a^2 \geq c_s^2 \\ \left(\dfrac{c_s^2 - c_a^2}{2c_a^2 + 2c_s^2}\right)\phi_3(m,\rho) + \left(\dfrac{c_s^2 + 3c_a^2}{2c_a^2 + 2c_s^2}\right)\psi\left(\left(c_a^2 + c_s^2\right)/2, m, \rho\right), & c_a^2 \leq c_s^2 \end{cases}$$

and $\mathrm{EW(M/M/m)} = \tau P(N \geq m)/m(1-\rho)$.

In the preceding equation, ψ and ϕ denote functions of m and ρ, found in Whitt (1993). Measures of interest other than expected waiting time, such as the number of busy servers (*E[B]*), the expected queue length (*E[Q]*), the expected number of customers in the system at any given time (*E[N]*), and the expected time in system (*E[T]*), can be found through simple analytical relationships between these measures and expected waiting time. These formulas are also presented in Whitt (1993) and summarized below.

$$E[N] = E[B] + E[Q]$$

$$E[T] = E[W] + \tau$$

$$E[B] = m\rho = \tau\lambda$$

$$E[Q] = \lambda E[W]$$

$$E[N] = \lambda E[T]$$

In order to measure the accuracy of the approximations, Whitt (1993) uses absolute difference and relative percentage error as performance metrics. As long as one of those values is below a designated threshold, Whitt contends the approximation is accurate. He defines an adjusted measure of error (*AME*) to combine the effects of the two performance measures:

$$AME = \min\{A|exact - approx.|, 100(|exact - appprox.|)/exact\}$$

where A is a constant determined by the user that weights the importance of the two performance metrics. Whitt does not present an exact value A for use in all calculations.

By evaluating the approximations based upon the performance metrics outlined above, Whitt (1993) is able to draw the conclusion that the approximations work best when the probability distributions for service time and inter-arrival time are not too irregular. The approximations are less accurate when c_a^2 and c_s^2 are large. The approximations overestimate the actual values both when m is large and ρ is small; however, the approximation works better than the old formula when traffic is heavy (ρ is large). Whitt compared the approximation for different queueing models. He concluded that the approximations work well for GI/M/m queues when $0< c_a^2<1$, as in an E_4/M/m model. It also performs well for E_2/E_2/m models but is not always better than the old approximation for G/H_2/m models. He identifies room for improvement in the approximations when $c_a^2>1> c_s^2$ and also when $c_a^2<1$.

2.1.3 Springer and Makens' Model

Springer and Makens (1991) evaluated five different approximations for the GI/G/1/k queue, which consists of single-station queues with finite buffers and general arrival and service processes. The results of this paper can also be applied to the more general M/M/1/k system by ignoring the second moments of the inter-arrival time and service time distributions. Basic formulas were presented for the mean number of items in the system (L_s), the probability of the system being empty (p_o), and the probability of the system being full (p_k), which are given below. The performance measures L_q, W_s, and W_q can be computed from these values.

$$L_s = \begin{cases} \dfrac{\rho}{1-\rho} - \dfrac{(k+1)\rho^{k+1}}{1-\rho^{k+1}} & (\rho \neq 1), \\ \dfrac{1}{2}k & (\rho = 1). \end{cases}$$

$$p_0 = \begin{cases} \dfrac{1-\rho}{1-\rho^{k+1}} & \rho \neq 1, \\ \dfrac{1}{k+1} & \rho = 1. \end{cases}$$

$$p_k = p_0 \rho^k$$

The accuracy of five variations of these formulas was evaluated in terms of their relative accuracy and relative bias. In terms of relative accuracy, the approximation developed by Gelenbe was superior for both L_s and p_o. The absolute relative error of Gelenbe's approximations was 0.0638 for L_s and 0.0294 for p_o, both at a significance level of 0.01. The Gaver-Shedler approximation was best for p_k, producing an absolute relative error of 0.2288. The Gelenbe model was biased when the squared coefficients of variation are both high and low, but overall it possessed less bias than the other approximations. For example, changing c_a^2 by 0.378 caused an overestimation of 1.42% in Gelenbe's model, but caused an overestimation of 22.28% in Yao and Buzacott's model. The approximations developed by Gelenbe for L_s and p_o and the approximation developed by Gaver-Shedler for p_k are presented below.

$$p_0 = \begin{cases} \dfrac{1-\rho}{1-\rho^2 e^{\gamma(k-1)}} & \rho \neq 1, \\ \dfrac{1}{2+2(k-1)/(c_a^2+c_s^2)} & \rho = 1. \end{cases}$$

$$L_s = \begin{cases} \dfrac{p_0\rho\left[\dfrac{1}{\gamma}-\dfrac{1}{2}\right]+p_k\left[k-\dfrac{1}{\gamma}-\dfrac{1}{2}\right]}{\rho-1}+p_k k & \rho \neq 1, \\ \dfrac{1}{2}k & \rho = 1. \end{cases}$$

$$p_k = \begin{cases} \dfrac{(1-p_0)(1-e^{\gamma})e^{\gamma(k-1)}}{1-e^{\gamma k}} & \rho \neq 1, \\ \dfrac{1-p_0}{k} & \rho = 1. \end{cases}$$

2.2 Neural Network Methods

The work most closely related to ours is that of Shin, Sargent, and Goel (2002). Shin, Sargent, and Goel present a systematic approach to queueing approximation via meta-modeling using Gaussian radial basis functions, which are specialized feed-forward networks having three layers that map an input space onto a desired output space, and use Gaussian basis functions. They use the SG algorithm to determine the number of basis functions, the basis function centers, and the basis function weights. For a detailed explanation of the SG algorithm, we refer the reader to Shin and Goel (1998, 2002). An additional parameter specifies the coverage of the input space, with 100% coverage requiring one basis function for every input value, permitting direct control over the model complexity. They demonstrate their approach on the M/M/1 queue, with a data set created by fixing the arrival rate and varying the service rate. They vary the basis function widths to create a variety of different meta-models for which they compute both the fitting and test mean squared errors (MSE) defined by the equation:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad (4)$$

They then show how each meta-model performs against the known closed-form equation for the expected wait time for the M/M/1 queue. Unlike the approach taken by Shin, Sargent, and Goel (2002), we use a custom neural network created by off-the-shelf software and tailored to our problem specifications. In addition, our approach uses back propagation instead of the SG algorithm to set the weights. When evaluating our approximation, we consider only test data sets, not fitting data sets, to get an idea of how valuable our approximation is in the real world. To that end, we consider average absolute relative error, instead of mean square error (which is an un-scaled error metric) because it is more relevant to real-world applications. Finally, we present an approximation for the GI/G/m for which there is no closed-form analytical model to test against, requiring simulation to obtain the "true" values for the expected wait time in queue. To reduce the time required to train our neural network to approximate the GI/G/m queue, we used Whitt's GI/G/m approximation as our starting point. We train our neural network on $\vec{x} = (MTBA, c_a^2, MST, c_s^2, m)$, as well as the output from Whitt's approximation, the second, third, and fourth moments of the arrival and service distributions, and an exact simulation model.

3 Solution Methods

Hybrid modeling (see Shanthikumar and Sargent [1983] and Sargent [1994]) involves the combination of analytic and simulation models to solve a problem. In this section, we present solution approaches that will be investigated within this research related to hybrid analysis of logistical supply chains. Our basic approach is similar to that of Whitt (1983) and of Sage and Sykes (1994). In such approaches, the network is decomposed into subcomponents to which the most appropriate modeling technique is applied.

In order to begin to develop such an approach, we must first investigate the models that will serve as the underlying analytical approximations within the network. To begin that effort, we are building upon the work presented in Whitt (1993) for the GI/G/m queue. Future work will investigate approaches to combining the analytical models within a network framework.

Our approach begins with the development of a better approximation for the GI/G/m queue. We will examine the building of robust models for the basic building blocks (G/G/s, G/G/s/N, etc.) needed to model the stochastic storage elements with application to supply chain performance analysis. While many approximations for these types of models already exist, our approach will be different in two ways. First, we are concerned with developing robust approximations. That is, they perform well over a wide variety of conditions and model assumption violations. Second, our approach to developing the approximations will be non-traditional in the sense that it will not follow standard stochastic process modeling practices. We utilize neural networks to enhance the development of the models.

To make this discussion more concrete, let us consider a specific approximation for the expected waiting time in a GI/G/s queue. One class of queueing approximations (see Whitt [1993]) involves characterizing the arrival process and the service process of the queue by the first two moments of the distributions. Let λ^{-1} be the mean and c_a^2 be the squared coefficient of variation for the inter-arrival distribution, and let τ and c_s^2 be the mean and the squared coefficient of variation for the service time distribution function. Approximations are then developed based upon the parameters $\vec{x} = (\lambda, c_a^2, \tau, c_s^2, s)$. This set of input parameters involves only second-moment

information plus the number of servers so the approximation is not a full specification of the queueing system.

Adapting the notation used in Segal and Whitt (1989), let W be the waiting time before beginning service and let $y = E[W]$ be its expected value. A simple functional approximation given in Segal and Whitt (1989) for $E[W]$ in the GI/G/s queue is

$$E[W](\lambda, c_a^2, \tau, c_s^2, s) = \left[\frac{c_a^2 + c_s^2}{2}\right] E[W]_{(m/m/S,\lambda,\tau)}$$

where $E[W]_{(m/m/S,\lambda,\tau)}$ represents the exact value computed from the M/M/s queue. In this case, an analytically tractable queueing model serves as the basis for a more robust approximation. Whitt (1993) contains a better, albeit more complicated, approximation than that given in the equation above. It should be clear that the mapping ability is limited for approximations based upon only the first two moments. There are a wide variety of more complicated approximations, but our point here is that analytical models can be used as building blocks. To get other performance measures such as the mean number in the queue, one can use conservation laws such as Little's formula or derive specific approximations for each performance measure.

Another way to build an approximating function for the expected waiting time would be to fit an induction model via some technique (least squares, non-linear regression, neural networks, group method of data handling (GMDH), etc., directly to the output of the system over a wide range of input parameters and distributions. In this case, we let $(\vec{x}_1, \vec{y}_1), \ldots, (\vec{x}_k, \vec{y}_k)$ be input/output pairs from the system where $\vec{x}_k$ represents the kth input and $\vec{y}_k$ the kth output and $\vec{y}_k = f(\vec{x}_k)$. Let us consider the response variable y_k univariate for simplicity of discussion, but these techniques are not limited to that case. While it is possible to build an induction model based upon observations of the actual system, it will be impractical due to the large amount of data required; however, an induction model could be built based upon input/output pairs from a detailed simulation. One can think of this approach as a form of simulation meta-modeling (see Friedman [1996], for example).

This process is illustrated in *Figure 3.1* and labeled Case 1 where W^s represents observations from the simulation, W^a represents the output of the queueing approximation, and $\hat{W}$ represents the expected waiting time based upon the induction model. The W^s entering the induction model from the bottom is meant to indicate that the induction model will be trained on observations of W^s. For Case 1, we have $\vec{x} = (\lambda, c_a^2, \tau, c_s^2, s, W^s)$ and $y = \hat{W}$ where we have dropped the subscript k for notational convenience. Case 2 illustrates the potential of using approximations in addition to the simulation to form $\hat{W}$. For Case 2, we have $\vec{x} = (\lambda, c_a^2, \tau, c_s^2, s, W^a, W^s)$ and $y = \hat{W}$. We will discuss this possibility further, but first we must discuss how approximations and models can be combined to form other approximations.

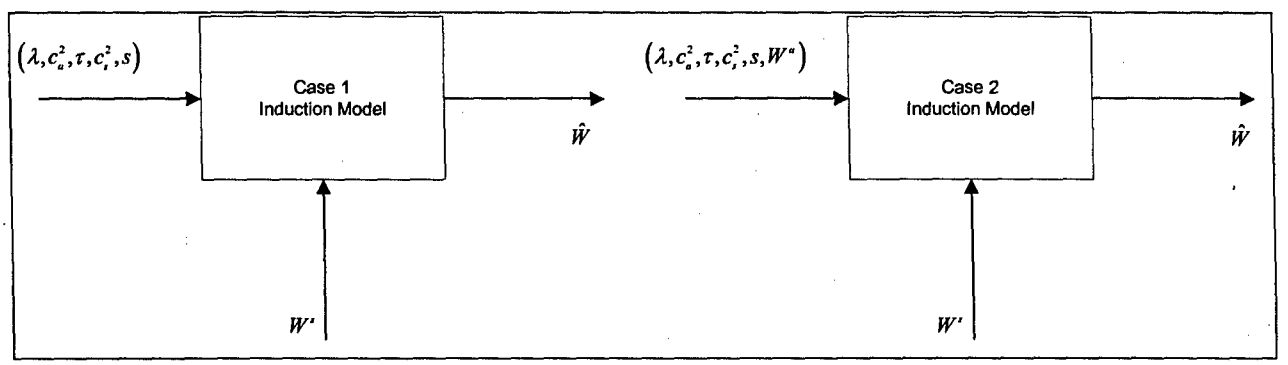

Figure 3.1: Induction Model Cases

Statistical learning networks, in particular neural networks, have a long history and are becoming increasingly accepted, provided one can discern the hype from the facts within the literature. For a historical perspective on neural networks, we refer the reader to, for example, Anderson and Rosenfeld (1988) or Hecht-Nielsen (1990). Given the development of backpropagation learning techniques, neural networks are now recognized as an important statistical modeling tool. In fact, Sarle (1994) shows artificial neural networks are "nothing more than non-linear regression and discriminant models that can be implemented with standard statistical software." Mathematical proofs (see Irie and Miyake [1988]; Hornik, Stinchcombe, and White [1989]; Cybenko [1989]; and Funahashi [1989]) have shown that neural networks can serve as universal approximators; that is, they can approximate any function to any desired degree of accuracy. Funahashi and

Nakamura (1993) have shown a continuous time-recurrent neural network can serve as a universal approximator for the finite time trajectory of any given dynamical system. These results indicate that a neural network can serve as a universal approximator. In practice, whether or not a good approximation is achieved is problematic.

Figure 3.2 illustrates a fully connected multi-layer feed-forward architecture for approximating the waiting time in a GI/G/s queue using M/M/s, M/D/s, and D/M/s as basis sub-models as in Kimura (1994). In *Figure 3.2*, the analytical models serve as inputs to the input layer of the network as well as moment information. This is probably the easiest approach to applying the induction approach to this problem. In this approach, the standard transfer functions can be used. In a sense, the network is "correcting" for the discrepancies in the approximate queueing models. Besides incorporating the queueing models as inputs, this methodology could allow the analytical models to be incorporated into the structure of the network by specifying them as acceptable transfer functions, that is, functions on the nodes in the network. In fact, Lemke (1997) describes an approach called "Active Neurons," which essentially allows the most appropriate transfer functions to be selected.

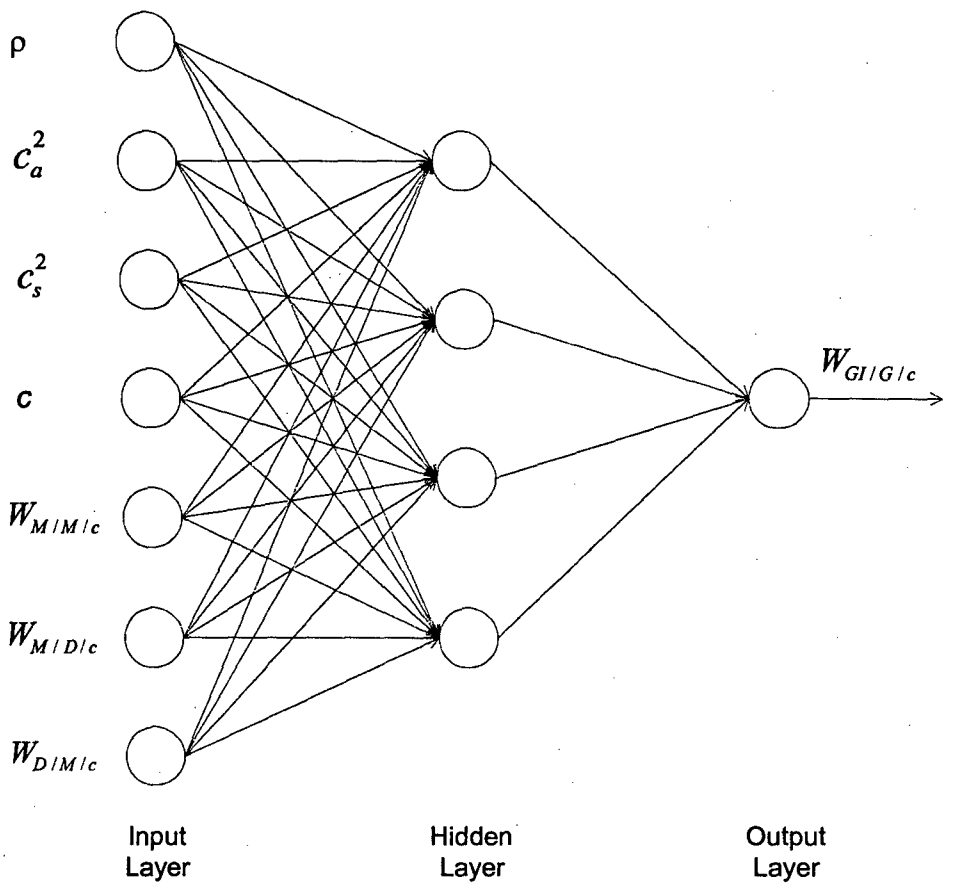

Figure 3.2: Multi-Layer Feed-forward Network

To summarize, our approach will be to investigate the use of neural networks to improve the approximations associated with the underlying analytical models. The approach taken to investigate this method will be discussed in *Section 4* of this document. To integrate the analytical models with simulation, we must consider methods to combine the analytical models with the logistical network simulation.

4 Developing the Single-Station Approximation

To build our single-station approximation for the GI/G/m, we begin with an existing approximation and train a neural network to correct the error with additional moment information, producing a better approximation.

4.1 Motivation

We chose the GI/G/m approximation from Whitt (1993) as the starting point for our neural network approximation. The model presented has unlimited waiting room, m identical parallel servers, and the queue operates on a first-come first-served basis. All service and inter-arrival times are independent, identically distributed random variables with general distributions specified by their first two moments. The arrival process is specified by λ and c_a^2, which are the arrival rate and the squared coefficient of variation of the inter-arrival time. The service process is specified by τ and c_s^2, which are the mean service time and the squared coefficient of variation of service time. Other assumptions made by Whitt (1993) are that traffic intensity, ρ, is less than 1, and that mean service time, τ, is normalized equal to 1. The traffic intensity, ρ, is equal to $\lambda\tau/m$.

In addition to the GI/G/m approximation in Whitt (1993), we used the following inputs for the neural network: $\left(MTBA, MST, c_a^2, c_s^2, m, W_q^a, W_q^s\right)$ where W_q^a and W_q^s are the expected waiting time values from Whitt's approximation and from the simulation, respectively. We then added the second, third and fourth moments of the arrival and service distributions as inputs. These additional moments provide a more accurate specification of the arrival and service distributions than is possible with Whitt's two-moment approximation.

4.2 Test Case Generation Methods

To train the neural network, we needed a set of test cases for which the wait time in queue was known. We developed and tested two test case generation methodologies. Each methodology was evaluated and compared in terms of its coverage of the neural network input parameter space. We discuss each method and its advantages and disadvantages in the following sections.

4.2.1 Distribution Parameter-Based Generation

The first test case generation mechanism is based upon directly generating the parameters of the distributions to be tested. The disadvantage of this method is that it is difficult to guarantee suitable distributional properties for the inputs of the queueing system, $(\lambda, \mu, c_a^2, c_s^2, m)$. To generate these test cases, we randomly select one of the six following distributions (independently) for both the arrival and service distributions: (exponential, uniform, triangular, gamma, Weibull, and lognormal). Once the distributions had been selected, values for the distribution parameters were selected based upon the information in *Table 4.1*. As indicated in the table, each parameter of the various distributions is constrained to fall within some given range of values.

Table 4.1: Generation Method 1 Distribution Parameters

Distribution	Parameters		
Exponential	λ ~uniform(1,25)		
Uniform	min~uniform(1,25)	max~uniform(min,25)	
Triangular	min~uniform(1,25)	max~uniform(min,25)	max~uniform(1,25)
Gamma	α ~uniform(1,5)	β ~uniform(1,5)	
Weibull	α ~uniform(1,25)	β ~uniform(1,25)	
Lognormal	E[X]~uniform(1,25)	$\sqrt{V[X]}$ logstdev~uniform(1,25)	

These parameters were used to compute λ, μ, and the offered load, $a = \lambda/\mu$, for the queue. The offered load and ρ (a randomly generated number from 0.05 to 0.95 inclusive) were then used to compute the required number of servers via $m = a/\rho$. *Table 4.2* presents sample summary statistics for test cases generated via this approach. *Table 4.3* presents the coverage across the distributions.

Table 4.2: Generation Method 1 Parameter Statistical Summary

	m	λ	μ	c_a^2	c_s^2
Mean	5.643	0.12239	0.12014	0.3505	0.3256
Stdev	9.171	0.12949	0.12416	0.8429	0.7228
Sample size	4994	4994	4994	4994	4994
Minimum	1	0.04	0.04002	0	0
Q1	2	0.05318	0.05332	0.0070	0.0066
Media	3	0.07577	0.07527	0.0675	0.0533
Q3	6	0.13107	0.13102	0.4529	0.4179
Maximum	119	1.07228	1.03293	22.0620	16.7383

Table 4.3: Generation Method 1 Distributional Coverage

Distribution	Arrival (%)	Service (%)
Exponential	16.3	16.3
Uniform	16.4	17.5
Triangular	16.3	16.7
Gamma	18.3	15.6
Weibull	16.4	17.2
Lognormal	16.4	16.7

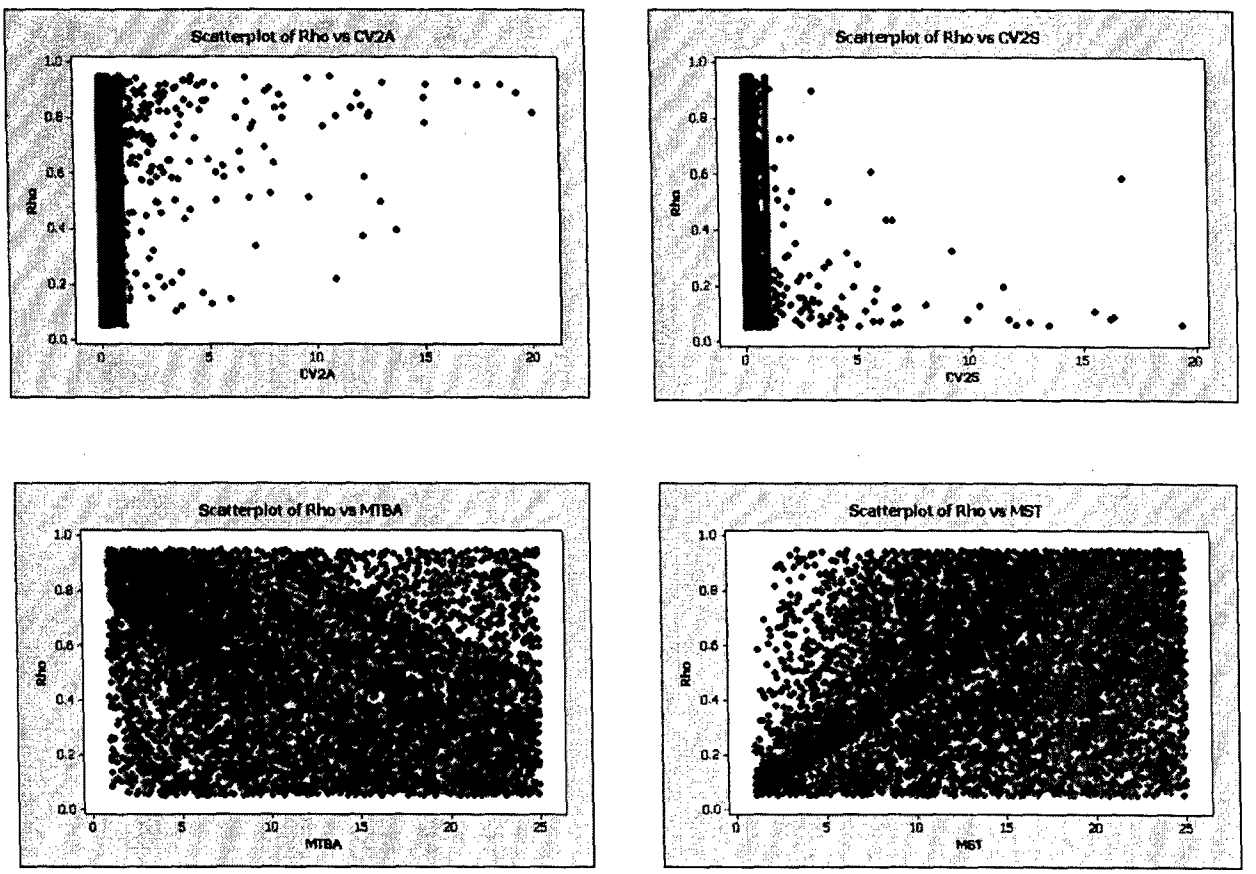

Figure 4.1: Generation Method 1 Parameter Scatter Plots

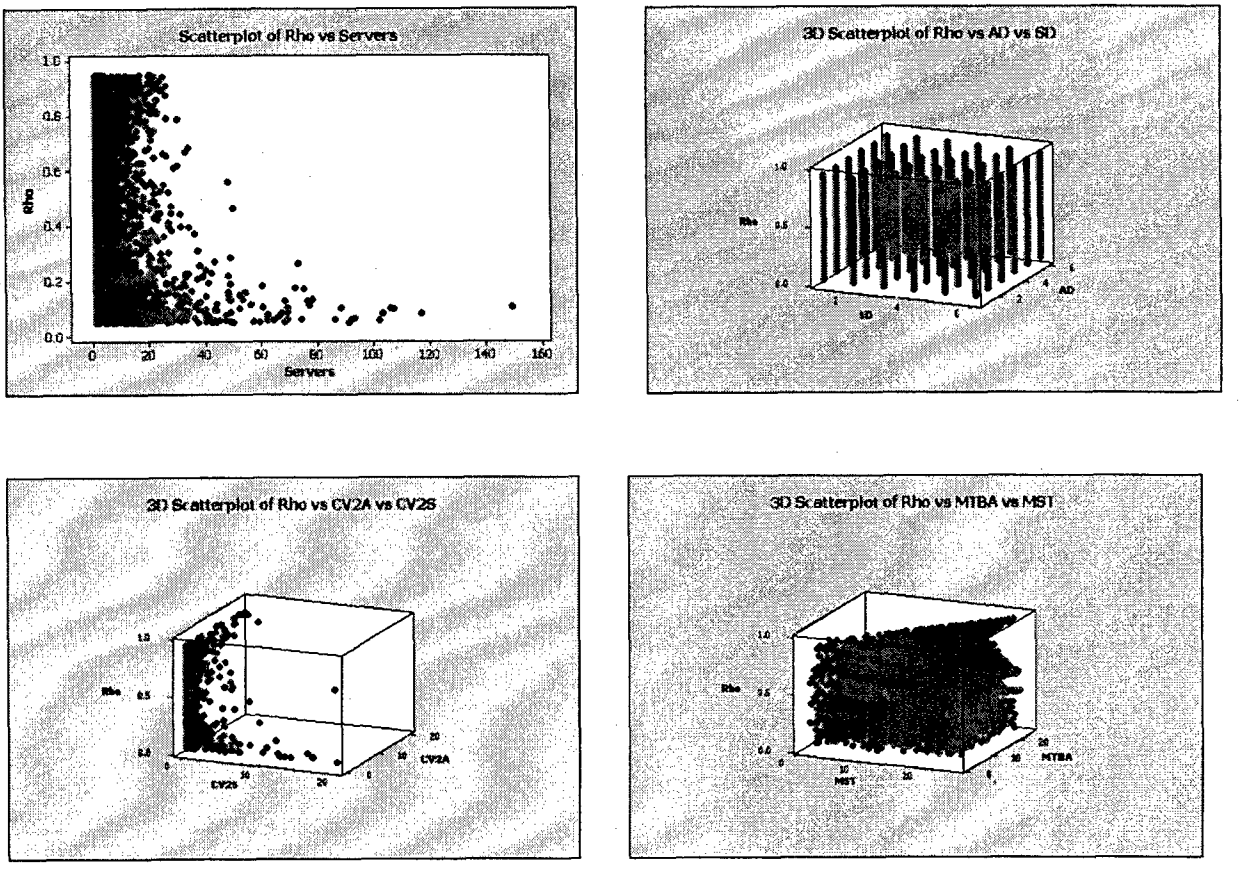

Figure 4.2: Generation Method 1 Parameter Coverage

As seen in *Figure 4.1* and *Figure 4.2*, test case generation Method 1 yields less-than-ideal coverage of the parameter space for the parameters that define the queue. In addition, it was difficult to determine how to change the range of the parameters of the underlying distribution to get a better coverage. We hypothesized that a more uniform coverage of the queue parameter space would give the neural network better information, thus improving its ability to learn the approximation. In addition, we wanted to cover a range that would prove useful in real-world problems. This motivated the development of an alternative test case generation method, which is discussed next.

4.2.2 Queue Parameter-Based Generation

The underlying approach associated with this method is to generate cases of the form $(MTBA, MST, c_a^2, c_s^2, m)$ that properly cover the parameter space. The difficulty with this method

is that the parameters of the distributions must then be derived. In addition, it will be difficult to guarantee the distributional properties of the parameters of the distributions.

The test case generation method is as follows. The mean time between arrivals ($MTBA = 1/\lambda$) is randomly chosen ~ Uniform(0,1). The number of servers was a randomly chosen integer ~ Uniform(1,150). The queue utilization, ρ, was chosen randomly ~ Uniform (0,1). The mean service time ($MST = 1/\mu$) was computed by the following equation: $MST = \rho m(MTBA)$. The result gives MST generated over the range (0,150).

Next, a distribution type is chosen for arrivals from the following six distribution types: (1) exponential, (2) uniform, (3) gamma, (4) Weibull, (5) triangular, and (6) lognormal. Each type is equally likely. Let K represent a random variable that indicates the six types of distributions. The method used for computing the distribution parameters depends upon the distribution chosen. Let X represent a random variable from the given distribution with $E[X]$ and $V[X]$ being its mean and variance, respectively. In addition, let c^2 represent the squared coefficient of variation, that is, $c^2 = V[X]/E^2[X]$. For arrivals, $E[X]$ = MTBA and for service $E[X]$ = MST. Note that all parameters are 0 unless otherwise noted. Thus, according to the above discussion:

$$MTBA \sim Uniform(0,1)$$
$$m \sim Discrete\ Uniform(1,150)$$
$$\rho \sim Uniform(0,1)$$
$$MST = \rho \times m \times MTBA$$
$$K \sim Discrete\ Uniform(1,6)$$

We now know the mean of the arrival and service distribution, that is, for arrivals $E[X]$ = MTBA and for service $E[X]$ = MST. We must now determine the other parameters of the distribution. In order to do this, we assume a viable range for the squared coefficient of variation when appropriate for the distribution and then randomly generate a value for the squared coefficient of variation uniformly over this range. From the squared coefficient of variation, we determine the variance of the distribution and then solve for the parameters of the distribution

using its mean and variance. Each distribution will have a set of equations that must be solved in order to match its parameters to the generated mean and variance.

Table 4.4 and *Table 4.5* present the equations for determining the parameters for each distribution. Notice that in the case of the Weibull distribution, the root of a non-linear equation must be solved. In the case of the triangular distribution, the shape of the distribution was limited to three cases: symmetric, right-skewed and left-skewed. This was done for two reasons. First, because the triangular has three parameters, we must assume an additional relationship to match the mean and variance to the parameters; thus, we assume that the minimum is equal to the mode, the maximum is equal to the mode, or that the mode is equal to the mean. Secondly, this allows us to specifically control the skewness.

Table 4.4: Generation Method 2 Distribution Parameters

Exponential MTBA = $E[X]$ ~ Uniform(0,1) for arrivals MST = $E[X] = \rho m(MTBA)$ for service
Uniform(a, b) $c^2 \sim Uniform(0,1/3)$ $V[X] = c^2 \times E[X]$ $a = E[X] - \sqrt{3 \times V[X]}$ $b = E[X] + \sqrt{3 \times V[X]}$
$Gamma(\alpha = shape, \beta = scale)$ $c^2 \sim Uniform(0,50)$ $V[X] = c^2 \times E[X]$ $\alpha = E[X] \times E[X] / V[X]$ $\beta = V[X] / E[X]$

$Lognormal(\mu_l, \sigma_l^2)$

$c^2 \sim Uniform(0,50)$

$V[X] = c^2 \times E[X]$

$\mu_l = E[X]$

$\sigma_l^2 = V[X]$

Table 4.5: Generation Method 2 Distribution Parameters

Weibull($\alpha = shape, \beta = scale$)

$$c^2 \sim Uniform(0,50)$$

$$V[X] = c^2 \times E[X]$$

$$solve: c = \sqrt{\frac{\Gamma\left(1+\frac{2}{\alpha}\right)}{\Gamma^2\left(1+\frac{1}{\alpha}\right)} - 1} \quad for\ \alpha$$

$$\beta = \frac{E[X]}{\Gamma\left(1+\frac{1}{\alpha}\right)}$$

Triangular(a=min, c=mode, b=max)

Randomly choose among one of three equally likely cases:

Symmetric

$$c^2 \sim Uniform(0,1/6)$$

$$V[X] = c^2 \times E[X]$$

$$a = E[X] - \sqrt{6 \times V[X]}$$

$$c = E[X]$$

$$b = E[X] + \sqrt{6 \times V[X]}$$

Positive Skew

$$c^2 \sim Uniform(0,1/8)$$

$$V[X] = c^2 \times E[X]$$

$$a = E[X] - \sqrt{2 \times V[X]}$$

$$c = a$$

$$b = E[X] + 2\sqrt{2 \times V[X]}$$

Negative Skew

$$c^2 \sim Uniform(0,1/2)$$

$$V[X] = c^2 \times E[X]$$

$$a = E[X] - 2\sqrt{2 \times V[X]}$$

$$c = b$$

$$b = E[X] + \sqrt{2 \times V[X]}$$

The distributional properties of the test cases are significantly improved over the previous test case generation methodology. *Table 4.6* presents sample summary statistics for test cases generated via this approach. *Table 4.7* presents the coverage across the distributions. *Figure 4.3* and *Figure 4.4* present the scatter plots for the distributional parameters. Note that while MTBA appears to be uniformly distributed, MST does not; however, the parameter space is still fairly well covered. In *Figure 4.3*, we have a scatter plot for the arrival distribution's squared coefficient of variation (CV2A). Note that it is nearly uniformly distributed with a higher concentration around 0 due to the exponential, uniform, and triangular distributions, all of which have squared coefficients of variation less than or equal to 1.0: The scatter plot for the squared coefficient of variation for the service distribution indicates a similar pattern.

Table 4.6: Generational Method 2 Parameter Statistical Summary

	m	MTBA	MST	c_a^2	c_s^2
Mean	75.271	0.50075	19.022	12.914	13.021
Stdev	43.419	0.29016	21.878	16.067	16.049
Sample size	7999	7999	7999	7999	7999
Minimum	1	0.00008	0.001	0	0
Q1	37	0.24113	3.030	0.223	0.238
Median	76	0.50665	10.484	1	1
Q3	113	0.75055	27.173	25.416	25.830
Maximum	150	0.99997	130.591	49.999	49.981

Table 4.7: Generation Method 2 Distributional Coverage

Distribution	Arrival (%)	Service (%)
Exponential (1)	16.0	16.7
Uniform (2)	16.2	16.2
Triangular (5)	17.3	16.2

Distribution	Arrival (%)	Service (%)
Gamma (3)	16.8	17.3
Weibull (4)	16.9	16.6
Lognormal (6)	16.7	17.0

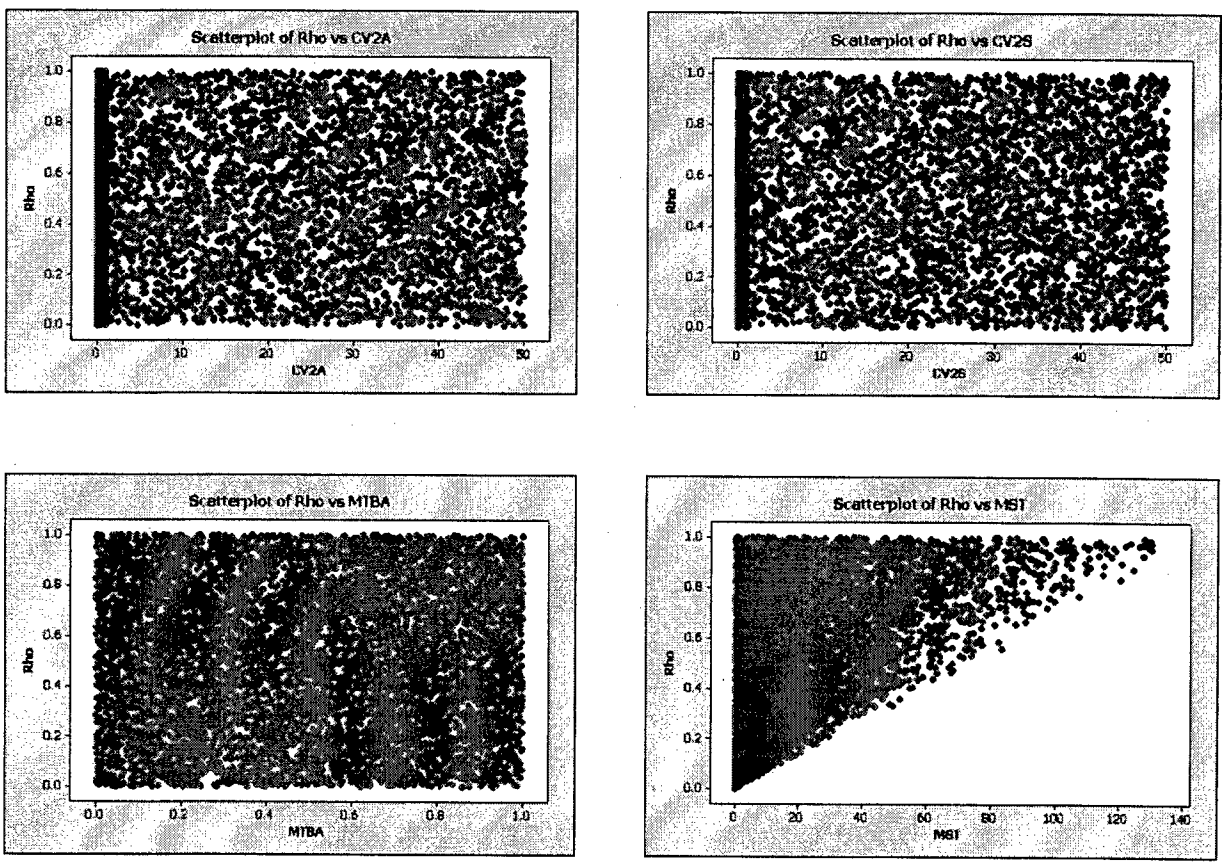

Figure 4.3: Generation Method 2 Parameter Scatter Plots

In the scatter plot of rho versus MST, we see that it appears to cover half its space nearly uniformly, divided by an upward-trending diagonal line. The cover is constrained in this way because MST is a function of both rho and the number of servers, both of which have finite bounds. For example, it is impossible to have a rho/MST combination of (0.2,100), because it would require the number of servers to be at least 500, which exceeds its range of (0,150). As

seen in *Figure 4.4*, the coverage of rho versus the number of servers is quite good. The distributions are uniformly distributed across their types as expected.

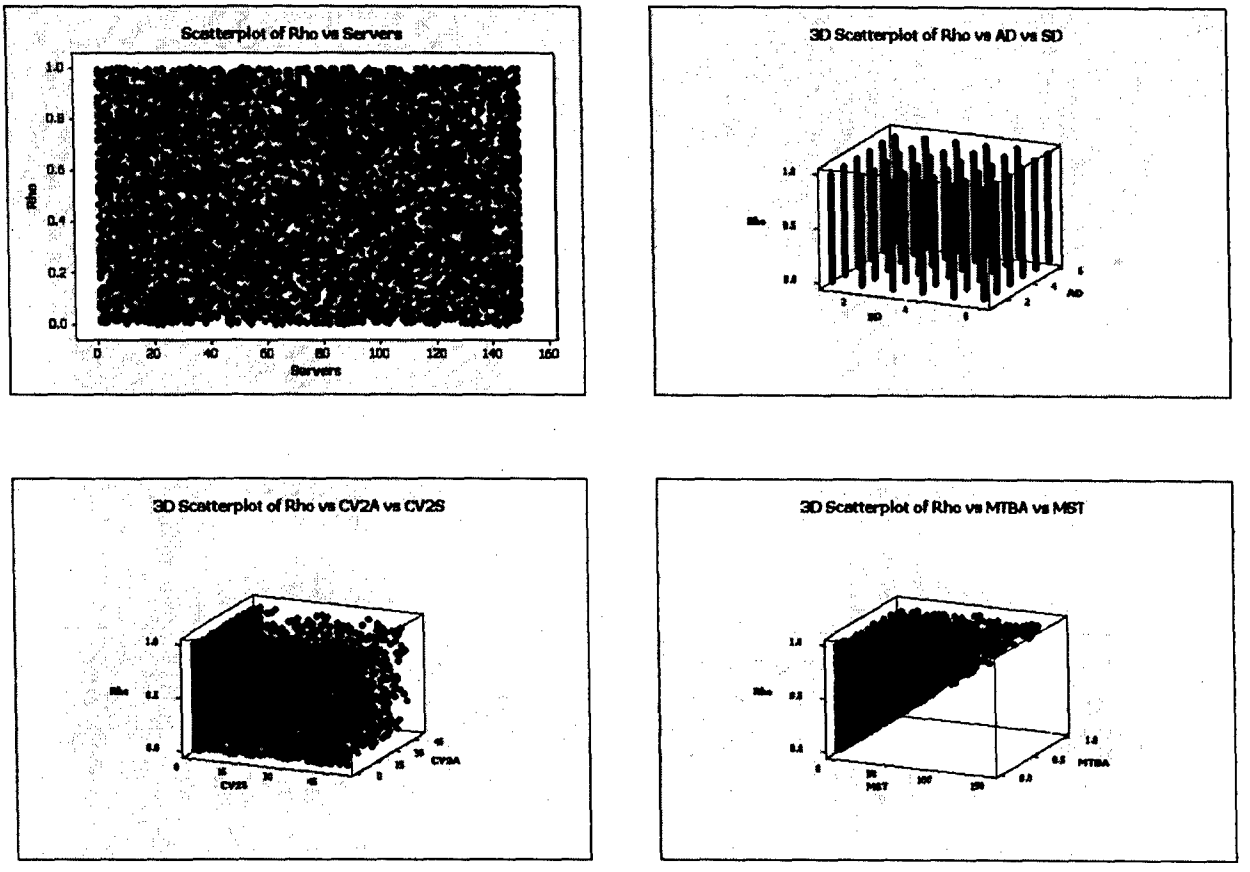

Figure 4.4: Generation Method 2 Parameter Coverage

Figures 4.5 – 4.8 show the parameter plots for the two data generation methods side-by-side to facilitate easy comparison of the two methods. As seen from the plots, Method 2 has better coverage in the squared coefficients of variation, servers, and mean time between arrivals. Method 1 appears to be a little better in mean service time; however, it should be noted that in Method 1, MST does not extend all the way to zero, unlike in Method 2. In addition, it is unclear whether MST would retain its near-uniformity if the underlying parameters of the service distribution were changed.

Method 1 **Method 2**

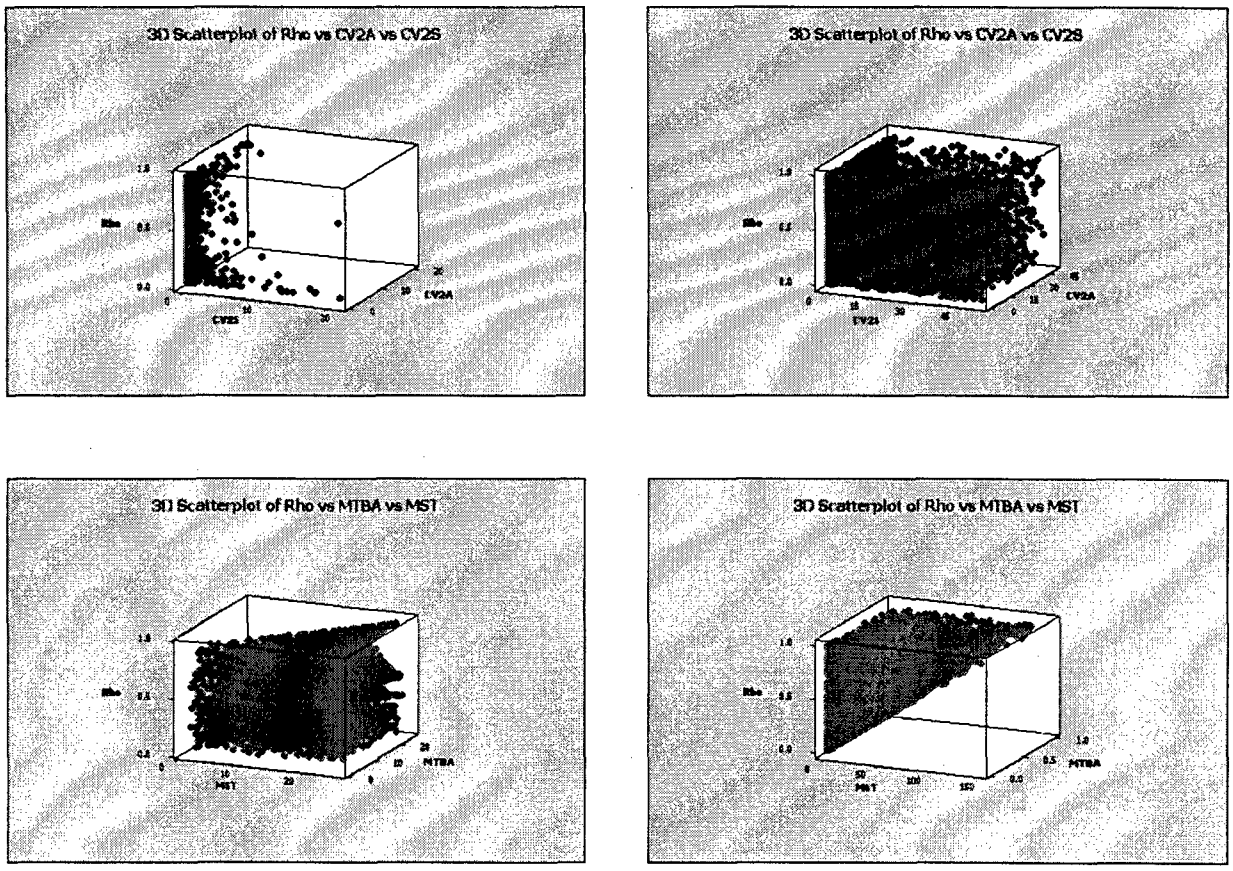

Figure 4.5: Comparison of Different Data Generation Methods*

*** MTBA, MST, CV2A and CV2S**

Method 1 **Method 2**

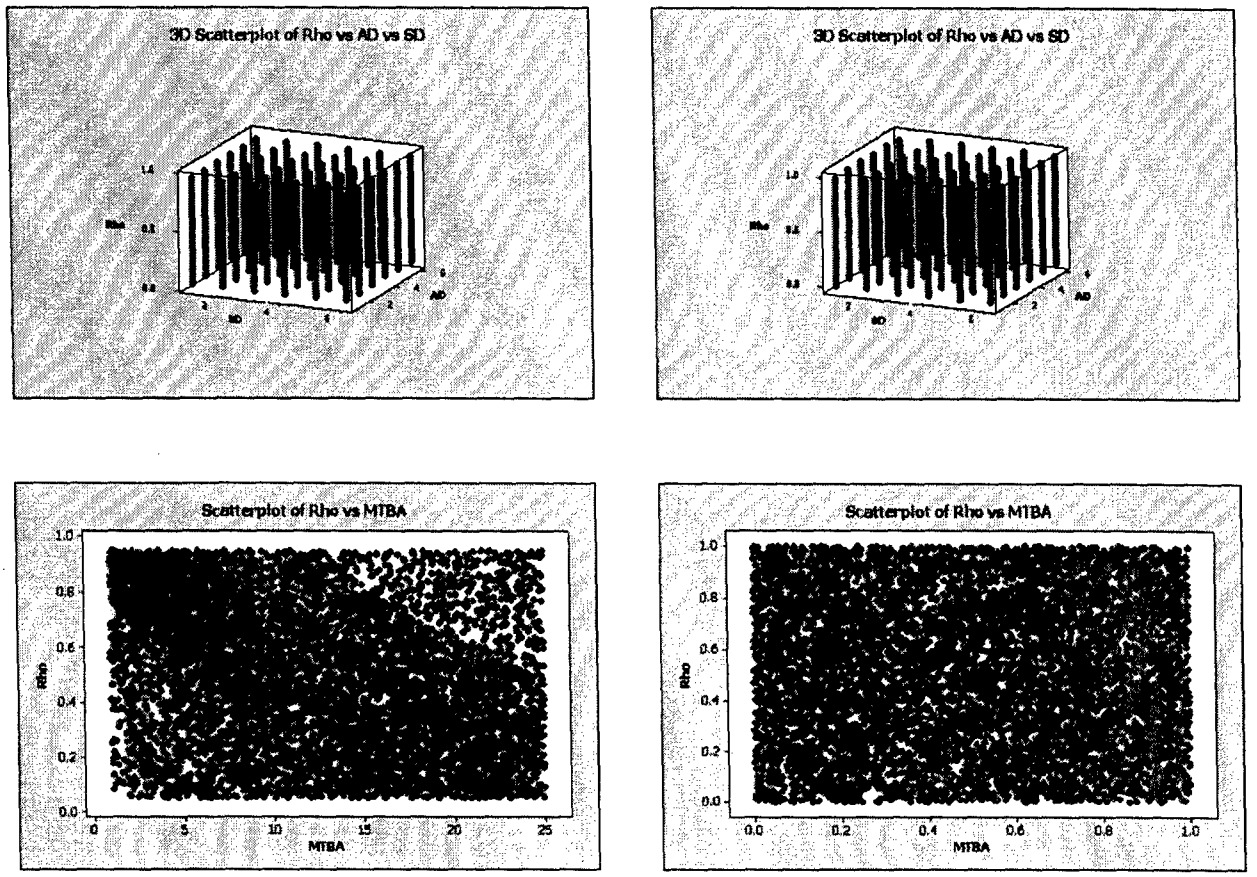

Figure 4.6: Comparison of Different Data Generation Methods**

** AD, SD and MTBA

Method 1 **Method 2**

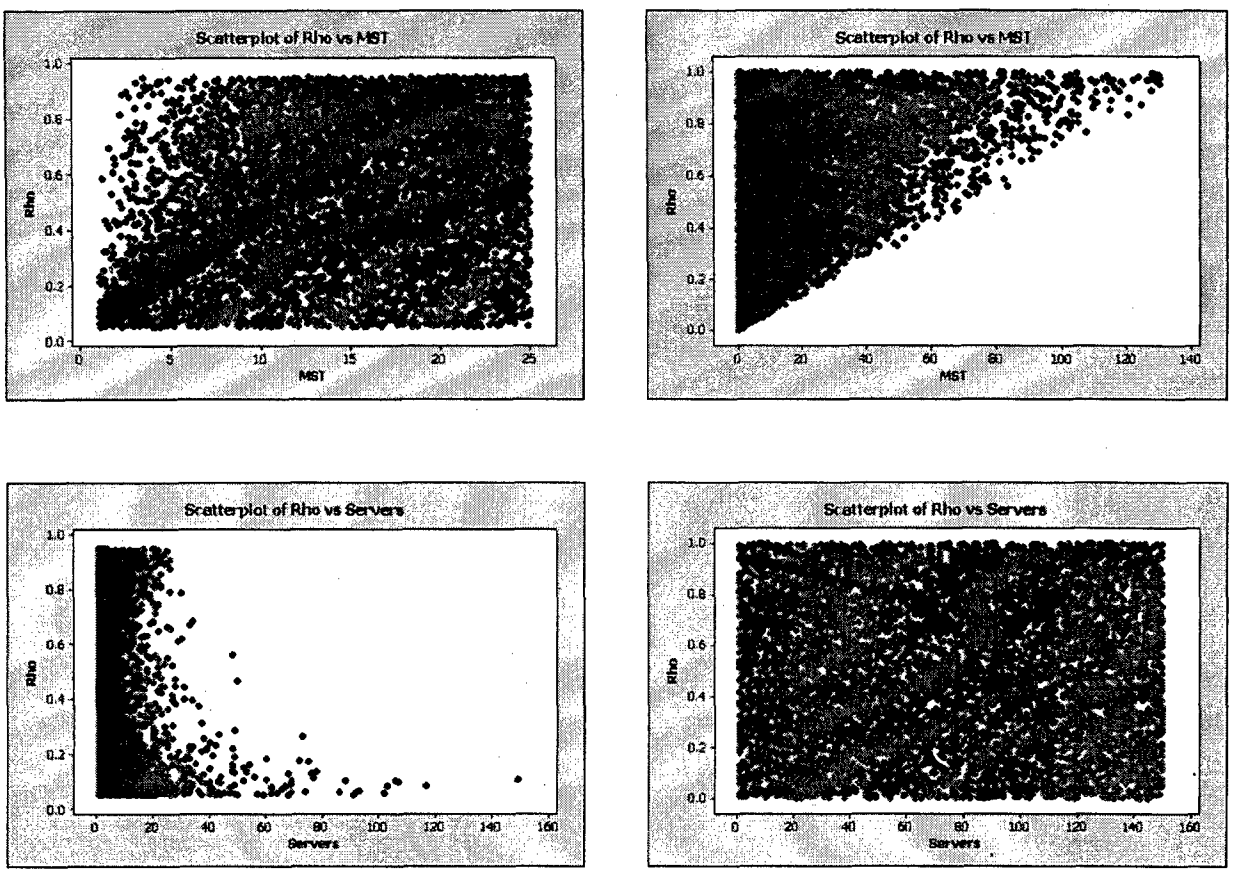

Figure 4.7: Comparison of Different Data Generation Methods***

***** MST and Servers**

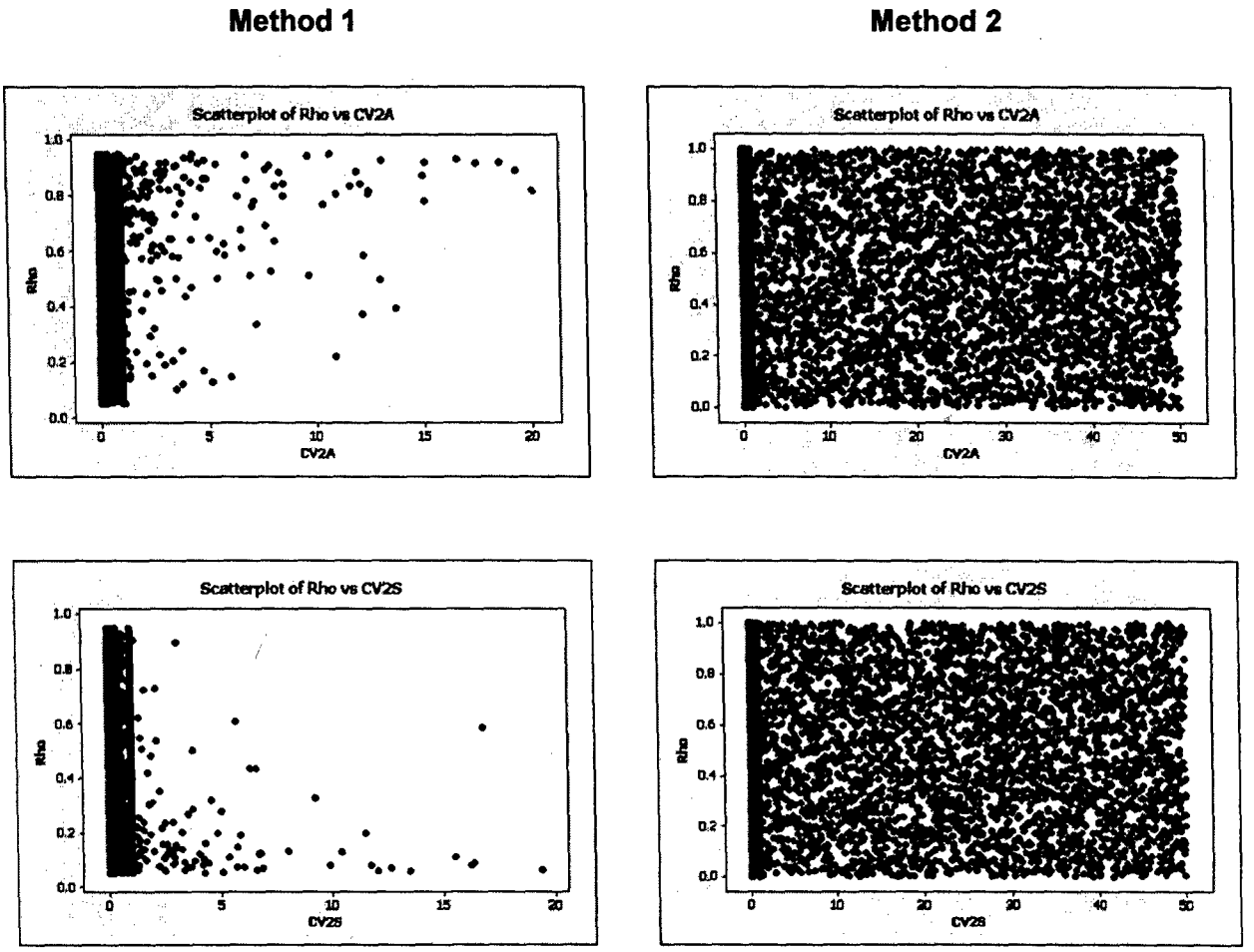

Figure 4.8: Comparison of Different Data Generation Methods****

****** CV2A and CV2S**

Based upon these results, we decided to use Method 2 for generating the test cases for the neural network training regiment. A total of 8000 test cases were generated. For each case, a simulation must be executed to estimate the performance of the queueing system. Then the cases can be used for training.

4.3 G/G/c Simulation

Initially, all simulations were developed using Arena®. Later, we changed the development to Java™ for a number of reasons, primarily due to difficulties we were having in implementing a half-width terminating condition within Arena®. A secondary reason was speed; Arena® proved very slow.

Finally, we experienced difficulties with the Gamma random generation algorithm within Arena®. Initially, we used half-width terminating conditions. This, however, resulted in very long simulation runs for certain test cases, so we implemented a maximum execution time limit that terminated a replication early if it had run for more than an allotted amount of real-world time. While this kept the simulation run-lengths down, it also greatly complicated verification of the simulation by making the simulation unrepeatable due to differences in execution speeds on different machines and even on the same machine under differing conditions. For these reasons, we went to a simple simulation-time-based warm-up and run-length. This provided repeatability but required us to determine a suitable run-length for each replication. We decided to use approximations from Whitt (1989).

The JSL's (Java™ Simulation Library) current version has packages that support random number generation, statistical collection, basic reporting and discrete-event simulation modeling. The development of a simulation model is based upon subclassing the ModelElement class that provides the primary recurring actions within a simulation (setup, before replication, warm-up, after replication, and end simulation) and event scheduling and handling. The user adds developed-model elements to an instance of Model and then executes the simulation.

The queueing simulation involves the development of a ModelElement that represents the queue, the server and the arrival/service processes. The class GGCWSModel, shown below, represents this object. In this class, we use response variables to collect the waiting time in the system and other performance measures.

Exhibit 4.1: GGc WSModel

```
public class GGcWSModel extends ModelElement {

    private Queue myWaitingQ;
    private int myNumServers;
    private DistributionIfc myServiceDistribution;
    private DistributionIfc myArrivalDistribution;
    private RandomVariable myServiceRV;
    private RandomVariable myArrivalRV;
```

```
    private TimeWeighted myNumBusy;
    private ResponseVariable myWs;

    private ArrivalListener myArrivalListener;
    private EndServiceListener myEndServiceListener;

    public GGcWSModel() {
        this(1, new Exponential(1.0), new Exponential(0.5));
    }

    public GGcWSModel(int numServers, DistributionIfc ad, DistributionIfc sd)
{
        setName("GGcWSModel");
        setNumberOfServers(numServers);
        setServiceDistribution(sd);
        setArrivalDistribution(ad);

        myWaitingQ = new Queue("GGC Q1");
        addModelElement(myWaitingQ);

        myNumBusy = new TimeWeighted(0.0, "NumBusy");
        addModelElement(myNumBusy);

        myWs = new ResponseVariable("WS");
        addModelElement(myWs);
        myWs.turnOnCountBasedStopping(1000);
        myWs.turnOnCountBasedWarmUp(100);

        myArrivalListener = new ArrivalListener();
        myEndServiceListener = new EndServiceListener();
    }
```

Exhibit 4.2: Event Routine Logic

```
    class ArrivalListener implements ActionListenerIfc {
        public void action(JSLEvent event) {
```

```
            QObjectIfc c = myWaitingQ.enqueue(new Double(getTime()));

            if (myNumBusy.getValue() < myNumServers){
                serveNextCustomer();
            }

            scheduleArrival();
        }
    }

    class EndServiceListener implements ActionListenerIfc {
        public void action(JSLEvent event) {
            myNumBusy.decrement();
            QObjectIfc c = (QObjectIfc)event.getMessage();
            double ws = getTime() - c.getTimeStamp();
            myWs.setValue(ws);
            if (myWaitingQ.size() > 0 ) {
                serveNextCustomer();
            }
        }
    }
```

A simulation model can be made by creating a model and attaching an instance of the GGcWSModel to the model. Then, the simulation can be run by creating an experiment and attaching the model to the experiment for execution.

Exhibit 4.3: Creating and Running a Model

```
            // create the containing model
            Model m = new Model();

            // create the model element and attach it to the main model
            GGcWSModel ggc = new GGcWSModel();
            m.addModelElement(ggc);

            // create the experiment to run the model
            Experiment e = new Experiment(m);
```

```
// set the parameters of the experiment
e.setNumberOfReplications(5);
e.setLengthOfReplication(11000.0);
e.setLengthOfWarmUp(1000.0);

// turn on the desired reporting
e.turnOnBatchReport();
e.turnOnExperimentReport();
e.turnOnReplicationReport();

// tell the experiment to run
e.runAll();
```

This process was automated by allowing the simulation's arrival and service processes to be specified by the generated test cases. The test cases were saved in a Microsoft® Access database and read in using Java's JDBC™ functionality.

4.4 Running the Test Cases

The following steps were performed for each case in the table containing the simulation test cases:

1. Read fields from database and use them to set the simulation parameters
2. Run the simulation
3. Write simulation results back out to the database

We set the maximum number of entities to 1,000,000 with a warm-up of 500,000 entities. In the exhibit below, we show the code contained within GGcWSModel used to set these limits.

Exhibit 4.4: Setting the Max Entities and Warm-up

```
myWs = new ResponseVariable("WS");
addModelElement(myWs);
myWs.turnOnCountBasedStopping(1000000);
myWs.turnOnCountBasedWarmUp(500000);
```

Exhibit 4.5: Capturing the Simulation Output

```
public class GGcWSModelObserver extends ModelElementObserver{
      private ResultSet myRS;

      public GGcWSModelObserver(ResultSet rs) {
            super();

            myRS = rs;
      }

      public void afterExperiment(ModelElement m, Object arg){
            GGcWSModel ggcModel = (GGcWSModel) m;

            double wqHW, lqHW, wsHW, wq, ws, wqVar, wsVar, lq, ls, lqVar,
lsVar, numBusy, warmup, runlength;
            long numArrived, numServed;

            wqHW = ggcModel.getWQHalfwidth();
            lqHW = ggcModel.getLQHalfwidth();
            wsHW = ggcModel.getWSHalfwidth();
            numArrived = ggcModel.getNumEntitiesArrived();
            numServed = ggcModel.getNumEntitiesServed();
            wq = ggcModel.getTimeInQAverage();
            wqVar = ggcModel.getTimeInQVariance();
            ws = ggcModel.getTimeInSystemAverage();
            wsVar = ggcModel.getTimeInSystemVariance();
            lq = ggcModel.getNumberInQAverage();
            lqVar = ggcModel.getNumberInQVariance();
            numBusy = ggcModel.getNumberBusyAverage();
            warmup = ggcModel.getLengthOfWarmUp();
            runlength = Math.min( ggcModel.getApproximateRunLength(),
65000000.0 );

            try {
                  myRS.updateDouble("WQ_HW," wqHW),
                  myRS.updateDouble("LQ_HW," lqHW);
                  myRS.updateDouble("WS_HW," wsHW);
```

```
                myRS.updateLong("NumArrived," numArrived);
                myRS.updateLong("NumServed," numServed);
                myRS.updateDouble("WQ,"wq);
                myRS.updateDouble("WS," ws);
                myRS.updateDouble("LQ," lq);
                myRS.updateDouble("varWS," wsVar);
                myRS.updateDouble("varWQ," wqVar);

                double approxWQ = myRS.getDouble("WhittApproxWQ");
                double approxWS = myRS.getDouble("WhittWS");
                double absError = Math.abs(ws - approxWS);

                myRS.updateDouble("WQ_Error," wq - approxWQ );
                myRS.updateDouble("WhittAbsError," absError);
                myRS.updateDouble("WhittAbsRelError," absError/ws);
                myRS.updateDouble("BusyServers," numBusy);
                      myRS.updateDouble("Warmup," warmup);
                myRS.updateDouble("RunLength," runlength);
                myRS.updateRow();
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
}
```

For the correlated arrivals cases, we generated a correlation coefficient ~ Uniform(0,1). We modified the simulation code as follows:

Exhibit 4.6: Setting the Arrival Distribution (Correlated Arrivals)

```
            myAD = getDistribution(myADnum, myADP1, myADP2, myADP3);
            mySD = getDistribution(mySDnum, mySDP1, mySDP2, mySDP3);


            double[] coefs = {myArrCorrelationCoef};
```

```
            myGGc.setArrivalDistribution(coefs, myAD);
            myGGc.setServiceDistribution(mySD);
            myGGc.setNumberOfServers(myServers);
```

```
      public void setArrivalDistribution(double[] c, DistributionIfc d) {

            if (d == null)
                  throw new IllegalArgumentException("Arrival Time
Distribution was null!");

            myArrivalDistribution = d;

            if (myArrivalRV == null){// not made yet
                  myArrivalRV = new ARTARandomVariable(c,
myArrivalDistribution);
                  myArrivalRV.setName("Arrival RV");
                  addModelElement(myArrivalRV);
            } else { // already had been made, and added to model
                  // just change the distribution
                  ((ARTARandomVariable)
myArrivalRV).setDistribution(myArrivalDistribution);
                  ((ARTARandomVariable) myArrivalRV).setCoefficients(c);
            }
      }
```

4.5 Validation/Verification

To validate the simulation model, we created a simple GI/G/m model manually in Arena®. We then selected 20 test cases, performing the same steps for each case:

1. Set up Arena® model with test case parameters
2. Run Arena® model simulation
3. Compare Arena® results with JSL simulation results

For the 20 test cases compared, our JSL simulation closely matched the output of the Arena® model. Note that in case numbers 2 and 5, there are large discrepancies between JSL and Arena®. Both cases use the gamma distribution, for which Arena® has a rarely occurring random variate generation error. For Case 2, we generated 100,000 random variates using Arena®, the JSL, and MINITAB®. For the JSL, we used two different methods: one method based upon numerical inversion of the cumulative distribution function, and the second method based upon an acceptance-rejection method. In *Table 4.8*, we present the descriptive statistics for each of these four methods. Case 2 has the shape parameter equal to 0.0266494848851034 and the scale parameter equal to 1.34756065089927 and is representive of the cases in which we found the problem with Arena®. For this case, this distribution was used as the service distribution. In general, Arena® tended to perform more poorly as the shape parameter approached zero. In typical simulation scenarios, it is doubtful that such low values for the shape parameter would be observed; however, in our context and due to the parameter coverage associated with our test case generation mechanism, such values are very likely. As one can see from the table, Arena's mean is too low, it has significantly higher variance, and there are too many observations in the tail of the distribution (notice the high 3rd quartile and the excessively high maximum). This is confirmed via the box plot given in *Figure 4.9*. Because Arena's algorithm has higher variance and produces more random variates in the tail of the distribution, we get longer-than-typical service times and thus more queueing than should be expected. This manifested itself in the results. Obviously, when it is important to control the error in fitting to the degree required in this research, such a difference cannot be tolerated.

Table 4.8: Descriptive Statistics for Gamma Testing

	Arena®	JSL Inversion	JSL A/R	MINITAB®
Count	100000	100000	100000	100000
Mean	0.0347	0.03527	0.03825	0.03659
StDev	0.2408	0.21783	0.22572	0.22714
Minimum	0.0000	0.00000	0.00000	0.00000
1st Quartile	0.0000	0.00000	0.00000	0.00000

Median	0.0006	0.00000	0.00000	0.00000
3rd Quartile	0.0079	0.000002	0.00002	0.00002
Maximum	31.4472	9.39469	6.92864	8.70346

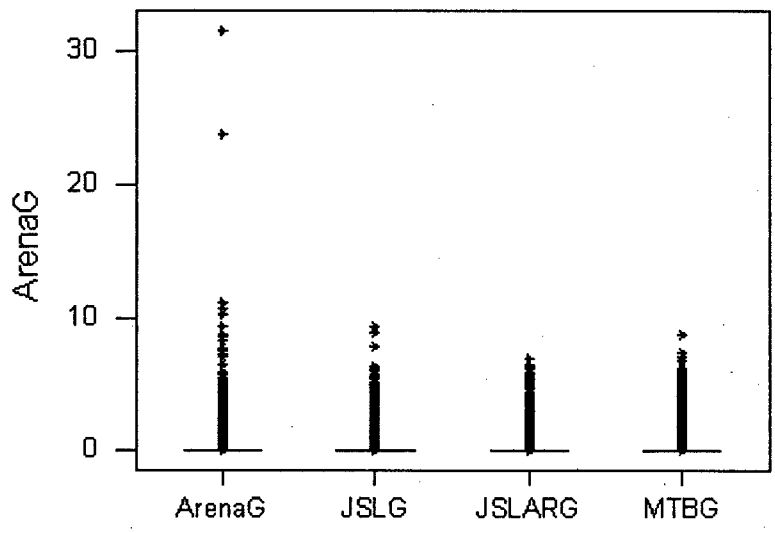

Figure 4.9: Box Plots for Gamma Testing

Thus we determined that the error lies with Arena® and that our simulation is correct for these cases.

Exhibit 4.7: Comparison of WQ Between JSL Simulation and Arena®

Case Number	JSL WQ	Arena® WQ
1	0.0	0.0
2	0.4722164637	0.00000228

Case Number	JSL WQ	Arena® WQ
3	0.0004895954	0.00070627
4	50.468557708	49.8721
5	0.1462066871	0.7197
6	7.1359898076	7.2072
7	0.0	0.0
8	0.0000002281	0.00000059
9	0.0	0.00178797
10	0.0	0.0
11	0.0020464259	0.00000031
12	0.0	0.0
13	0.0	0.0
14	0.0000073843	0.00000445
15	9.9532776996	9.0675
16	0.0	0.0
17	0.0	0.0
18	0.1247152412	0.1177
19	0.0007551607	0.00083055
20	0.0	0.0

5 Developing and Testing the Neural Network Approximation

In this section, we discuss the software packages we used and the experiences with both. We also summarize our approach, describe how we implemented Whitt's approximation, and cover cases with independent arrivals and correlated arrivals.

5.1 Software Decision

For the majority of the project, we used a commercial software package called NeuroSolutions 4.0 from NeuroDimension. We continually worked to train the neural network to create a better GI/G/m queueing approximation than Whitt's approximation. Several months after using this software and after going through several cycles of modifying data generation methods and simulation techniques, we discovered what appears to be a significant and troubling flaw in the NeuroSolutions software. When training a neural network with NeuroSolutions, one specifies a desired output that the neural network is intended to approximate as closely as possible. When testing is performed, the software creates an output file formatted with two columns: desired output and actual output (the output of the neural network). The desired output column should have contained the same data that was fed into the neural network and on which the network was supposed to have been trained. There should be no discrepancy between what is being read into the software and what is being used to train the network; however, as *Table 5.1* shows, it was off, often quite significantly, from what had been read into the software. In particular, note the highlighted row. It became very clear that we could never achieve the desired error approximations under these conditions. This discovery called into question the entire software package and prompted us to search for a replacement package with very little time remaining in the project. As an additional note, this software problem caused significant delay in the project (approximately three months). We tried many different methods to improve our approximations. These efforts distracted significantly from the results we were able to achieve in the time allotted for the project.

Table 5.1: Example of Discrepancies Between WQ (In) and WQ (Out)

Des WQ (In)	Des WQ (Out)
0.4450841170	0.4451450000
0.0006873430	0.0008440000
56.3773346900	56.3772810000
0.1757049280	0.1755260000
7.1561559070	7.1560730000
0.0021201930	0.0021100000
0.0000017338	0.0000000000
8.5581994170	8.55817400000
0.1042025480	0.1042190000
0.0008299960	0.0008440000
0.0349463810	0.0350210000
0.3072925160	0.3071710000
0.0006944270	0.000844000
0.2741871870	0.2742600000
0.0578146150	0.0578060000
0.0000063257	0.0000000000

We found a package by NeuralWare™ called NeuralWorks Predict®, and downloaded a demo version. Even with the limit of 512 cases for training (of which only 340 were used for fitting with the remaining used for testing), the software was immediately able to tie Whitt's approximation within statistical error, something that had eluded us with NeuroSolutions for several months. In addition, the training times for Predict were a very small fraction of those required for NeuroSolutions. Finally, there are a number of different training options and customizations available in Predict that we have not yet tried. In what follows, we discuss our approach to training the neural networks.

5.2 Overview of Approach

To begin the training process, we first exported a data set from Microsoft® Access to a Microsoft® Excel spreadsheet format. Next, we opened that spreadsheet in Excel and ran the Predict add-in. We selected part of the data set on which to train, selected the inputs and desired output, configured a number of software parameters, and instructed the software to train the neural network. The inputs we used included (for both the arrival and service distributions):

- Mean
- Squared coefficient of variation
- Percentiles (0.1, 1st, 5th, 95th, 99th, 99.9th)
- Quartiles (1st and 3rd)
- Moments (2nd, 3rd, 4th)

Other inputs included:

- Servers
- Rho
- Upper bound on WQ from (Kleinrock [1976])
- Correlation coefficient (for correlated cases only)

To test the neural network after training, we ran it on a different range of test cases. We then computed statistics for the following error metrics for both Whitt's approximation and our neural network approximation: absolute error and absolute relative error. In order to get the values for Whitt's approximation, it was necessary to implement his equations.

5.3 Implementing Whitt's Approximation

We implemented Whitt's approximation for the expected waiting time in a GI/G/m queue in Java™. The code was modeled after the equations found in Whitt (1993). The method, which returns WQ for Whitt's approximation is shown below.

Exhibit 5.1: WQ from Whitt's Approximation

```
public double getApproxWaitInQ(){
    double ewqMMm = MMcWq(myRho, 1.0, myNumServers);
    double scv = (mySCVA + mySCVS)/2.0;
    double phi = phi(myRho, mySCVA, mySCVS, myNumServers);
    double wq = phi*scv*ewqMMm;

    return(myMST*wq);
}
```

Exhibit 5.2: Functions Required by GetApproxWaitInQ

```
public static double MMcWq(double rho, double mu, int c){
    if ((rho <= 0.0) || (rho >= 1.0))
        throw new IllegalArgumentException("Utilization must be (0,1)");

    if (mu <= 0.0)
        throw new IllegalArgumentException("Service rate must be > 0");

    if (c <= 0)
        throw new IllegalArgumentException("Number of servers must be >
0");

    double a = c*rho;
    double csa = erlangC(c, a);
    double cmwq = 1.0/((1.0 - rho)*c*mu); // conditional mean wait in
queue
    return(csa*cmwq);
}

private double phi(double rho, double scva, double scvs, int m){
    double phi,
    double scv = (scva + scvs)/2.0;
    if (scva == scvs){ // this case should reduce directly to psi
```

```
            phi = psi(scva, m, rho);
        } else if (scva > scvs){
            double d = 4.0*scva - 3.0*scvs;
            double a = (4.0*(scva - scvs))/d;
            double b = scvs/d;
            phi = a*phiOne(m, rho) + b*psi(scv, m, rho);
        } else {// scva < scvs
            if (scva == 0.0) // handle special case where equation 2.25
doesn't reduce
                phi = phiThree(m,rho);
            else {
                double d = 2.0*(scva + scvs);
                double a = (scvs - scva)/d;
                double b = (scvs + 3.0*scva)/d;
                phi = a*phiThree(m, rho) + b*psi(scv, m, rho);
            }
        }
        return(phi);
    }

    private double psi(double scv, int m, double rho){
        double psi;
        if (scv >= 1.0)
            psi = 1.0;
        else {
            double x = 2.0*(1.0 - scv);
            psi = Math.pow(phiFour(m, rho), x);
        }
        return(psi);
    }
```

Exhibit 5.3: More Functions Required by getApproxWaitInQ

```
    private double gamma(int m, double rho){
```

```
        double x;
        x = (1.0-rho)*(m-1.0)*(Math.sqrt(4 +5*m) - 2.0)/(16.0*m*rho);
        double gamma = Math.min(0.24,x);
        //System.out.println("gamma = " + gamma);
        return(gamma);
    }

    private double phiOne(int m, double rho){
        double phiOne = 1.0 + gamma(m, rho);
        //System.out.println("phiOne = " + phiOne);
        return(phiOne);
    }

    private double phiTwo(int m, double rho){
        double phiTwo = 1.0 - 4.0*gamma(m, rho);
        return(phiTwo);
    }

    private double phiThree(int m, double rho){
        double x = Math.exp(-2.0*(1.0-rho)/(3.0*rho));
        double phiThree = phiTwo(m, rho)*x;
        return(phiThree);
    }

    private double phiFour(int m, double rho){
        double x = (phiOne(m, rho) + phiThree(m, rho))/2.0;
        double phiFour = Math.min(1.0, x);
        return(phiFour);
    }
```

Exhibit 5.4: More Functions Required by getApproxWaitInQ

```
    public static double erlangB(int m, double a){
        double b = 1.0;
        double d = 1.0;
        double t = 1.0;
```

```
        for(int i=1; i<=m; i++){
            t = a*b;
            d = i + t;
            b = t/d;
        }

        return(b);
    }

    public static double erlangC(int m, double a){
        double b = erlangB(m,a);
        double t = m*b;
        double d = m - a*(1.0-b);
        return(t/d);
    }
```

5.4 Independent Arrivals Case

In this section, we present comparisons between the neural network and Whitt's GI/G/m approximation for cases with independent arrivals.

In addition to comparing our neural network approximation with Whitt's approximation, we also compare both approximations with mean service as an approximation of WS. This serves as a quick litmus test on the applicability of each approximation. The reasoning behind this decision is simple; both our neural network approximation and Whitt's approximation yield the expected waiting time in queue as outputs. To obtain an approximation for the waiting time in the system, we use the equation $WS = WQ + MST$. There are two components to the mean waiting time in system: mean waiting time in queue, and mean service time. Clearly, if we can better predict the mean waiting time in system with the mean service time alone than we can if we add an approximation for the mean time in queue, this implies that the approximation for WQ has no practical value. Since the mean service time is better and is a known quantity requiring no computation, we may as well use it and not expend limited resources approximating WQ. Thus, we always compare each approximation to the mean service time to determine whether we not

only did better than the other approximation, but also whether our approximation is at least as good as the mean service time.

To measure the performance of our neural network approximation, we use the following metrics: absolute error and absolute relative error, as defined in *Table 5.2*.

Table 5.2: Error Metric Definitions

Error Metric	Definition
Absolute Error	$\left\|W_q^s - W_q^a\right\|$
Absolute Relative Error	$\left\|\frac{W_q^s - W_q^a}{W_q^s}\right\|$

The superscript s denotes "simulation," while the superscript a denotes "approximation." Subscripts are used to differentiate between time in queue and time in system. For example, MST AE is given by

$\left|W_s^s - MST\right|$ and MST ARE is given by $\frac{\left|W_s^s - MST\right|}{W_s^s}$.

We initially trained the neural network using all test cases, including those where WQ was zero. We then hypothesized that if we could improve the neural network fit by training only on cases where WQ is non-zero, then, assuming we could accurately predict which cases were going to have no queueing, we would simply use MST for WS in those cases. For all others, we would use WQ + MST. This should eliminate a source of error, since when WQ is zero, in steady state WS = MST, and therefore attempting to estimate WQ, will likely only introduce error in the estimation of WS. Thus, we decided to test our hypothesis by splitting the experiment into two categories: data sets that excluded cases where WQ is zero, and data sets that did not.

5.4.1 Non-Zero WQ Only

We trained the neural network on 2500 test cases, and tested it on the remaining cases in the data set (approximately 2400).

Table 5.3: Comparison of ARE for Independent Cases (Non-Zero WQ)

	NN ARE	Whitt ARE	MST ARE
Mean	0.08925	0.1138	0.19838
StDev	0.25016	1.4016	0.29613
Minimum	0	0	0
Q1	0.00516	0.0019	0.00253
Median	0.02102	0.0109	0.02188
Q3	0.09080	0.1037	0.31904
Maximum	5.68965	68.1421	0.99919
$\hat{p}(0.01)$	0.35905	0.48916	0.41785
$\hat{p}(0.05)$	0.64387	0.65930	0.57298
$\hat{p}(0.10)$	0.76456	0.74520	0.63094
$\hat{p}(0.50)$	0.97540	0.97873	0.81735
$\hat{p}(1.00)$	0.99416	0.99625	1.00000
$\hat{p}(5.00)$	0.99917	0.99958	1.00000
95% CI on Mean (Lower)	0.07923	0.0577	0.18652
95% CI on Mean (Upper)	0.09927	0.1699	0.21024

From *Table 5.3*, notice that both the mean and variance of ARE for our approximation are lower than for the Whitt's approximation, as well as for MST. In addition, note that our third quartile and maximum values are much lower than Whitt's, implying that our approximation has better worst-case performance than Whitt's approximation. Examining the 95% confidence interval

bounds, notice that our approximation has a much tighter bound (due to a much lower variance), which means our approximation should perform more reliably than Whitt's. While the ARE for our approximation varies across a 2% range (at a 0.05 level of significance), Whitt's approximation varies across an 11.2% range. In this table, we define $\hat{p}(\gamma)$ as an estimate of the probability that the absolute relative error will be less than or equal to γ. These values provide insight into the cause behind differences in the mean and standard deviation of the ARE between the neural network and Whitt's approximation (or MST). For example, as is the case in *Table 5.3*, if Whitt has higher p-"hat" values than the neural network for increasing gamma, but the neural network has a lower sample mean, this suggests that the difference in sample means (and standard deviations) is due to the large difference in maximum error values. A paired t-test between the difference in the mean ARE for our approximation and Whitt's yields a p-value of 0.196, or slightly greater than 80% confidence that our approximation has a lower mean absolute relative error than Whitt's approximation.

Table 5.4: Comparison of AE for Independent Cases (Non-Zero WQ)

	NN AE	Whitt AE	MST AE
Mean	41.6	51.3	58
StDev	1265.3	1357.9	1270.9
Minimum	0	0	0
Q1	0.1	0	0
Median	0.2	0.2	0.3
Q3	1.4	2	5.3
Maximum	61473.1	61473.5	61477.6
95% CI on Mean (Lower)	0	0	7.1
95% CI on Mean (Upper)	92.3	105.7	108.9

Examining *Table 5.4*, we see similar results as with the absolute relative error. Note that the mean AE of our approximation is lower than Whitt's, and that our standard deviation is lower, resulting in a tighter 95% confidence interval. Performing a paired t-test, we get a p-value of

0.172, which means that we have 82% confidence that the mean absolute error of our neural network approximation will be less than the mean absolute error of Whitt's.

5.4.2 All WQ Cases Included

We trained the neural network on 5000 test cases and tested it on 2500 test cases. When we included the cases where WQ is zero, we saw an improvement in the neural network, as seen in *Table 5.5* below. Whitt's approximation also did better, but our approximation still had a smaller sample mean than Whitt's.

Table 5.5: Comparison of ARE for Independent Cases (WQ Can Be Zero)

	NN ARE	Whitt ARE	MST ARE
Mean	0.0733	0.0877	0.11965
StDev	0.4424	1.3877	0.24499
Minimum	0	0	0
Q1	0.0019	0.0006	0.00068
Median	0.0072	0.0034	0.00405
Q3	0.0421	0.0300	0.06796
Maximum	13.4297	68.1421	0.99919
$\hat{p}(0.01)$	0.56623	0.67387	0.63105
$\hat{p}(0.05)$	0.76951	0.78511	0.73029
$\hat{p}(0.10)$	0.84514	0.83754	0.77351
$\hat{p}(0.50)$	0.97839	0.98479	0.89196
$\hat{p}(1.00)$	0.99480	0.99600	1.00000
$\hat{p}(5.00)$	0.99880	0.99880	1.00000
95% CI on Mean (Lower)	0.0560	0.0333	0.11004
95% CI on Mean (Upper)	0.0907	0.1422	0.12926

Notice the difference between the means for our approximation and Whitt's is less than it was for the non-zero cases. Also, note that our standard deviation is not as low (although it is still much lower than Whitt's); however, our maximum value is much smaller than Whitt's maximum value, suggesting better worst-case performance for our approximation. Comparing our p-"hat" values, we see the same phenomenon as with the non-zero cases. A paired t-test on the difference of means between our approximation's ARE and Whitt's ARE yields a p-value of 0.307, meaning that we are less confident that our approximation has a lower mean ARE than Whitt's approximation when we included cases where WQ is zero; however, it has a lower maximum value and a tighter 95% confidence interval. Finally, note that the 95% confidence intervals of Whitt's approximation and MST overlap. A paired t-test shows a p-value for the difference between the mean absolute relative error of Whitt's approximation and MST of 0.123, while the p-value for the same test between the neural network approximation and MST is less than 0.0001. Thus, there is cause for concern that over the parameter space used, Whitt's approximation may not always outperform the mean service time as an estimator of the waiting time in the system, in terms of mean absolute relative error.

Table 5.6: Comparison of AE for Independent Cases (WQ Can Be Zero)

	NN AE	Whitt AE	MST AE
Mean	37.4	44.7	45.7
StDev	1244.1	1328.1	1247.2
Minimum	0	0	0
Q1	0	0	0
Median	0.1	0	0
Q3	0.4	0.4	0.8
Maximum	61474.7	61473.5	61477.6
95% CI on Mean (Lower)	0	0	0
95% CI on Mean (Upper)	86.2	96.8	94.7

In *Table 5.6*, we see similar results as for absolute relative error. Again, our approximation has a lower mean and standard deviation, as well as a tighter 95% confidence interval. The paired t-test on the difference in means between the neural network and Whitt's approximation shows a more favorable p-value of 0.234, or a 76% confidence level that the mean absolute error of the neural network is less than the mean absolute error of Whitt's approximation. A paired t-test between the neural network and mean service time shows a p-value of less than 0.0001, while the same test between Whitt's approximation and mean service time shows a p-value of 0.458. Thus again, there is cause for concern that Whitt's approximation may fail to improve over MST when used to approximate the expected wait in system.

5.4.3 Comparison of Methods

We now examine which of the two training methods is best. *Table 5.7* below shows a side-by-side comparison between the two methods for absolute relative error.

Table 5.7: Comparison for ARE Between Non-Zero WQ and All WQ

	Non-Zero WQ			All WQ		
	NN ARE	Whitt ARE	MST ARE	NN ARE	Whitt ARE	MST ARE
Mean	0.08925	0.1138	0.19838	0.0733	0.0877	0.11965
StDev	0.25016	1.4016	0.29613	0.4424	1.3877	0.24499
Minimum	0	0	0	0	0	0
Q1	0.00516	0.0019	0.00253	0.0019	0.0006	0.00068
Median	0.02102	0.0109	0.02188	0.0072	0.0034	0.00405
Q3	0.09080	0.1037	0.31904	0.0421	0.0300	0.06796
Maximum	5.68965	68.1421	0.99919	13.4297	68.1421	0.99919
95% CI on Mean (Lower)	0.07923	0.0577	0.18652	0.0560	0.0333	0.11004
95% CI on Mean (Upper)	0.09927	0.1699	0.21024	0.0907	0.1422	0.12926

From *Table 5.7* above, you can see that while the mean ARE is lower for the neural network, it is also much lower for Whitt's approximation. In addition, the neural network's standard deviation

has increased, while the standard deviation for Whitt's approximation has decreased. Performing hypothesis tests on the difference in means sheds more light on the influence the training method has on the relative performance of the neural network approximation. *Table 5.8* below shows p-values for the paired t-tests for the difference in the mean between the variable in a given row and the variable in a given column. The null hypothesis used in all cases was "less than 0," and the mean of the null hypothesis is zero.

Table 5.8: p-value Comparison Matrix

	Non-Zero WQ			All WQ		
	NN ARE	Whitt ARE	MST ARE	NN ARE	Whitt ARE	MST ARE
NN ARE		0.196	0.000		0.307	0.000
Whitt ARE			0.002			0.123

As seen in *Table 5.8*, training and using the network only on cases where WQ is non-zero (queueing occurs) provides the best results in terms of p-value or confidence that our approximation is better than Whitt's in terms of absolute relative error for the independent arrivals cases.

Table 5.9: Comparison for AE Between Non-Zero WQ and All WQ

	Non-Zero WQ			All WQ		
	NN AE	Whitt AE	MST AE	NN AE	Whitt AE	MST AE
Mean	41.6	51.3	58	37.4	44.7	45.7
StDev	1265.3	1357.9	1270.9	1244.1	1328.1	1247.2
Minimum	0	0	0	0	0	0
Q1	0.1	0	0	0	0	0
Median	0.2	0.2	0.3	0.1	0	0
Q3	1.4	2	5.3	0.4	0.4	0.8

	Non-Zero WQ			All WQ		
	NN AE	Whitt AE	MST AE	NN AE	Whitt AE	MST AE
Maximum	61473.1	61473.5	61477.6	61474.7	61473.5	61477.6
95% CI on Mean (Lower)	0	0	7.1	0	0	0
95% CI on Mean (Upper)	92.3	105.7	108.9	86.2	96.8	94.7

Table 5.9 shows comparisons between non-zero WQ and all WQ for absolute error. As you can see from the table, all three means (neural network, Whitt and MST) are lower for training on all WQ than for training on only non-zero WQ; however, the raw difference between them is smaller. For example, for the neural network trained on non-zero WQ, the mean absolute error for the neural network is nearly ten less than the mean absolute error for Whitt's approximation. For the neural network trained on all WQ, the mean absolute error for the neural network is 7.3 less than the mean absolute error for Whitt's approximation. Likewise, the standard deviations and 95% confidence interval upper bounds are higher across the board for the neural network trained only on non-zero WQ than for the neural network trained on all WQ. Also of note, the third quartiles of all three are dramatically smaller for the neural network trained on all WQ.

Table 5.10: p-value Comparison Matrix

	Non-Zero WQ			All WQ		
	NN AE	Whitt AE	MST AE	NN AE	Whitt AE	MST AE
NN AE		0.172	0.000		0.234	0.000
Whitt AE			0.257			0.458

Table 5.10 shows the p-values resulting from paired t-tests on the difference between the variable from a given row and the variable of a given column. The null hypothesis for the paired t-test is "less than 0." From the table above, we can see that we have a higher level of confidence now that the neural network approximation has a lower mean absolute error than Whitt's approximation when trained and tested only on non-zero WQ cases. This seems to agree with

Table 5.9, which we noted showed smaller differences among neural network, Whitt, and MST for the data set where WQ can equal zero than for the data set where WQ must be non-zero.

These results appear to agree with the results for absolute relative error, suggesting that we can expect better neural network performance by excluding cases from training and testing where WQ is zero. This relies heavily upon the ability of the practitioner to predict at the time of application whether WQ will be zero. At least for independent arrivals, Whitt's approximation for WQ can be used for this purpose. An appropriate cutoff value must be chosen such that acceptable levels of Type I and II errors are maintained. For any cases in which Whitt's approximate WQ falls below the cutoff, the actual WQ is assumed to be zero and MST is used to estimate WS. If Whitt's approximate WQ is at or above the cutoff, the neural network approximation is applied. More investigation is needed to determine an optimal or near-optimal strategy for setting the cutoff level. Another approach to identifying zero-queueing cases without simulation could utilize classifier neural networks specifically trained to identify cases where WQ is zero. In both cases, care would have to be taken so that error introduced in this stage does not negate the improvements obtained over Whitt's approximation in the training stage.

5.5 Correlated Arrivals Case

The primary focus of queueing approximation research has been for the case of renewal input processes and infinite capacity queues, specifically GI/G/s queues. Because logistic systems will carry a variety of items that circulate throughout the network, we may expect to have dependence between inter-arrival times, between service times, and between inter-arrival and service times. Much of the literature that has examined dependency within queueing networks has focused on communication systems. We refer the reader to Fendick et. al. (1989), Liew (1994), Cidon et. al. (1993), and Takine et. al. (1994) for more information on the dependence within queueing network systems. Parametric-decomposition approaches to queueing networks analyze the nodes in the network separately after making adjustments to the internal flow processes so they appear as renewal inputs to the individual queues. Then the current GI/G/s approximations can be used to analyze the individual nodes. Much research has been done to better represent the internal flow processes as renewal process (see, for example, the work by Albin [1984]). Another

would be to build more robust approximations for G/G/s and G/G/s/N queues that can better handle the dependencies found in these types of networks.

Tin (1985) reported results that indicate the mean queue lengths for correlated and uncorrelated arrivals can differ by a factor of nine in queueing systems with Markov-dependent arrivals. This result sparked interest by Patuwo et al. (1993), who examined the effect of correlation in an MR/M/1 queue as compared to the M/M/1 queue. In examining the effect of lag-1 correlation, they found the mean number in the system uniformly increases with increasing values of the correlation. They found that "even small correlations of up to 0.4 can cause the mean number in the system to nearly double"(Patuwo [1993]) To examine the effect of higher-order lags, they used the index of dispersion (IDI) (see Fendick et al. [1989]), which describes the cumulative correlation for the first inter-arrival times, service times, and cross-correlations between inter-arrival and service times. Again, the results indicated serious effects due to correlation. In Livny et al. (1993), the authors again study the effect of correlation on single-server queues with a variety of correlated arrivals and service demands based upon TES processes (see Melamed [1991]). They conclude that positive correlations always lead to performance degradation in terms of mean waiting times. For example, they simulated a single-server queue with utilization factor of 0.50 and positive lag-1 autocorrelation in the arrival process of 0.85. When compared to the benchmark case of an M/M/1 under independence assumptions, the mean waiting time was more than 200 times larger for the correlated case. The performance degraded even more significantly for high utilizations. Simcoe and Pei (1995) conclude that the common assumption of the uniform distribution model for traffic is inappropriate, because it can seriously underestimate the amount of output loading that occurs within the node. This raises serious questions for the ad hoc use of GI/G/s queueing approximations and suggests that the development of better G/G/s and G/G/s/N approximations is needed. It is for these reasons that we decided to examine and develop a neural network approximation for the correlated arrivals case.

In this section, we present comparisons between the neural network and Whitt's GI/G/m approximation for cases with correlated arrivals. We split our experiment with correlated arrivals into two classes: data sets that exclude cases where WQ is zero, and data sets that did not.

5.5.1 Non-Zero WQ Only

We trained the neural network on 5000 test cases and tested on the remainder of the data set (350 test cases). For correlated cases, training on only non-zero WQ cases introduced an unexpected problem. While you can see from *Table 5.11* and *Table 5.12* below that we had no problem beating Whitt's approximation, when comparing our approximation for WS to simply the mean service time (MST), we see that we did worse, statistically speaking. The reason for this anomaly is that whenever WQ is non-zero and the neural network returns zero (or a negative value is truncated to zero), we incur a very large absolute relative error for that case, demonstrably larger than we would if we simply said WS equals MST.

Table 5.11: Comparison of ARE for Correlated Arrivals (Non-Zero WQ)

	NN ARE	Whitt ARE	MST ARE
Mean	0.1388	0.5316	0.11992
StDev	0.7374	3.4111	0.22345
Minimum	0.0001	0	0
Q1	0.0045	0.0022	0.00191
Median	0.02808	0.0116	0.00921
Q3	0.1007	0.1297	0.11532
Maximum	13.4287	55.9119	0.99868
$\hat{p}(0.01)$	0.37536	0.48424	0.507163
$\hat{p}(0.05)$	0.63897	0.65903	0.670487
$\hat{p}(0.10)$	0.74785	0.71920	0.736390
$\hat{p}(0.50)$	0.93696	0.91117	0.914040
$\hat{p}(1.00)$	0.99713	0.94269	1.000000
$\hat{p}(5.00)$	0.99713	0.97708	1.000000
95% CI on Mean (Lower)	0.0612	0.1725	0.09639

	NN ARE	Whitt ARE	MST ARE
95% CI on Mean (Upper)	0.2164	0.8908	0.14344

From *Table 5.11*, we see that our approximation ARE has a much lower mean and standard deviation that Whitt's approximation. A paired t-test between the neural network approximation and Whitt's approximation yields a p-value of 0.017, giving a better than 98.3% confidence level that the mean absolute relative error of our approximation is less than the mean absolute relative error of Whitt's approximation. In addition, the third quartile and maximum for our approximation is less than for Whitt's approximation, suggesting better worst-case performance for our approximation for correlated arrivals. This is supported by the p-"hat" values in the table. At 10% absolute relative error and above, the neural network has a larger fraction of cases "in-range" than Whitt's approximation. Whitt's approximation has more cases that fall within $\pm$ 5% or less than our neural network approximation, but our approximation has fewer extreme values, resulting in better overall performance; however, neither our approximation nor Whitt's is statistically better than MST. The p-value for the paired t-test between our approximation and MST is 0.687, while for Whitt's approximation, it is 0.988. Thus we have clearly done better than Whitt, but the challenge remains to obtain a good approximation for the correlated case.

Table 5.12: Comparison of AE for Correlated Arrivals (Non-Zero WQ)

	NN AE	Whitt AE	MST AE
Mean	182.0	203.7	182.8
StDev	3290.4	3297.5	3290.6
Minimum	0	0	0
Q1	0.1	0	0
Median	0.2	0.1	0.1

	NN AE	Whitt AE	MST AE
Q3	0.9	1.4	1.1
Maximum	61472.8	61473.5	61477.6
95% CI on Mean (Lower)	0	0	0
95% CI on Mean (Upper)	528.4	550.8	529.2

Examining *Table 5.12* shows similar results as with ARE for cases with non-zero WQ. A paired t-test between the neural network and Whitt's approximation yields a p-value of 0.039, or a 96.1% confidence level that the mean absolute error of the neural network approximation is less than the mean absolute error of Whitt's approximation. A similar comparison to MST yields a surprising result, yielding a p-value of less than 0.0001, meaning that in terms of absolute error, our neural network gives a good approximation for correlated arrivals.

5.5.2 All WQ Cases Included

We trained the neural network on 5000 test cases and tested it on 2500 test cases. We saw an improvement when we included the cases where WQ is zero. As seen in *Table 5.12* and *Table 5.13* below, training all cases gave an improvement across the board; however, we were still unable to do statistically better than MST for an approximation for WS, but we have far from exhausted all possible directions for improving the neural network. Among the different possibilities, we are looking into bounding the output on the neural network, as there is a large penalty incurred by outputting values that are either too small or too large.

Table 5.13: Comparison of ARE for Correlated Arrivals (WQ Can Be Zero)

	NN ARE	Whitt ARE	MST ARE
Mean	0.09527	0.492	0.093376
StDev	0.30909	4.435	0.208278
Minimum	0	0	0
Q1	0.00220	0.001	0.000863

	NN ARE	Whitt ARE	MST ARE
Median	0.00791	0.004	0.004498
Q3	0.05484	0.047	0.045672
Maximum	7.14033	135.055	0.999589
$\hat{p}(0.01)$	0.545818	0.628651	0.631453
$\hat{p}(0.05)$	0.741497	0.755502	0.757503
$\hat{p}(0.10)$	0.807923	0.811124	0.806323
$\hat{p}(0.50)$	0.950380	0.930372	0.927171
$\hat{p}(1.00)$	0.990796	0.958784	1.000000
$\hat{p}(5.00)$	0.999200	0.983593	1.000000
95% CI on Mean (Lower)	0.08314	0.318	0.085206
95% CI on Mean (Upper)	0.10739	0.67	0.101546

In *Table 5.13*, we again see a much lower mean and standard deviation with the neural network approximation than with Whitt's approximation. In addition, our approximation has a much lower maximum absolute relative error than Whitt's approximation, as seen in the table. As with the non-zero WQ cases, the p-"hat" values suggest the neural network approximation has generally lower absolute relative error. In addition, the consistently greater p-"hat" values for increasing ARE suggest the neural network also has better worse-case performance than Whitt's approximation. A paired t-test between both approximations yields a p-value less than 0.0001, which means that we are very confident the mean absolute relative error of our neural network approximation is less than the mean absolute relative error of Whitt's approximation. A similar comparison to MST yields a far less favorable p-value of 0.629.

Table 5.14: Comparison of AE for Correlated Arrivals (WQ Can Be Zero)

	NN AE	Whitt AE	MST AE
Mean	54.7	81.6	48.7
StDev	1476.9	1557.2	1385.5
Minimum	0	0	0
Q1	0	0	0
Median	0.1	0.1	0
Q3	0.6	0.8	0.6
Maximum	61471.7	61473.5	61477.6
95% CI on Mean (Lower)	0	116.5	0
95% CI on Mean (Upper)	16.4	146.7	103.0

From *Table 5.14*, we see a similar result as we did for the non-zero WQ cases. Our approximation has a smaller mean, standard deviation, third quartile, and maximum than Whitt's approximation. Paired t-tests show our approximation is both better than Whitt's approximation and MST (p-values of 0.001 and <0.0001 were obtained, respectively). Thus, in terms of mean absolute error, our approximation not only outperforms Whitt's approximation for correlated cases, but it is also a good approximation.

5.5.3 Comparison of Methods

In this section, we compare the two training/testing methods for our neural network approximation to determine which is best for our purposes.

As seen in *Table 5.15* below, the mean and standard deviation for the neural network absolute relative error is lower for the data set containing all cases than it is for the data set containing only non-zero WQ cases. While Whitt's mean is lower, its standard deviation is higher for the "All WQ" data set. All three (the neural network, Whitt and MST) have lower first and third quartile, median, and maximum values for the "All WQ" data set than for the data set containing

only non-zero WQ cases. From this table, it is hard to determine conclusively which method is better, although "All WQ" appears to be the winner.

Table 5.15: Comparison for ARE Between Non-Zero WQ and All WQ

	Non-Zero WQ			All WQ		
	NN ARE	Whitt ARE	MST ARE	NN ARE	Whitt ARE	MST ARE
Mean	0.1388	0.5316	0.11992	0.09527	0.492	0.093376
StDev	0.7374	3.4111	0.22345	0.30909	4.435	0.208278
Minimum	0.0001	0	0	0	0	0
Q1	0.0045	0.0022	0.00191	0.00220	0.001	0.000863
Median	0.02808	0.0116	0.00921	0.00791	0.004	0.004498
Q3	0.1007	0.1297	0.11532	0.05484	0.047	0.045672
Maximum	13.4287	55.9119	0.99868	7.14033	135.055	0.999589
95% CI on Mean (Lower)	0.0612	0.1725	0.09639	0.08314	0.318	0.085206
95% CI on Mean (Upper)	0.2164	0.8908	0.14344	0.10739	0.67	0.101546

Table 5.16 shows more clearly the difference between the two neural network training/testing strategies. The p-values shown are for paired t-tests between row and column variables with a null hypothesis of "less than 0."

Table 5.16: p-value Comparison Matrix

	Non-Zero WQ			All WQ		
	NN ARE	Whitt ARE	MST ARE	NN ARE	Whitt ARE	MST ARE
NN ARE		0.017	0.687		0.001	0.629
Whitt ARE			0.988			1.000

As seen in *Table 5.16*, the "All WQ" method provides greater confidence that the mean absolute relative error of our neural network approximation is less than the mean absolute relative error of Whitt's approximation (99.9% versus 98.3%). In addition, we have greater confidence in the quality of our approximation, 37.1% that the mean ARE for our neural network is less than the mean ARE for MST for the "All WQ method" versus 31.3% for the "Non-Zero WQ" method. Also, note that the "All WQ" method has a higher p-value for the test between Whitt's approximation and MST, indicating that not only does our approximation perform better, but also that Whitt's approximation performs more poorly.

It is difficult to draw conclusions from *Table 5.17* below. The mean and 95% confidence limit upper bound for the neural network, Whitt's approximation, and MST all appear to be lower for the "All WQ" case than for the "Non-Zero WQ" case by roughly the same amount. Relative to the neural network and MST, Whitt's standard deviation rises when going from "Non-Zero WQ" to "All WQ," although all three have lower standard deviations for the "All WQ" method than the "Non-Zero WQ" method. The first and third quartile, median, and maximum values are also all lower for the "All WQ" method than for the "Non-Zero WQ" method.

Table 5.17: Comparison for AE Between Non-Zero WQ and All WQ

	Non-Zero WQ			All WQ		
	NN AE	Whitt AE	MST AE	NN AE	Whitt AE	MST AE
Mean	182.0	203.7	182.8	48.1	71.8	48.7
StDev	3290.4	3297.5	3290.6	1385.2	1460.6	1385.5
Minimum	0	0	0	0	0	0
Q1	0.1	0	0	0	0	0
Median	0.2	0.1	0.1	0.1	0	0
Q3	0.9	1.4	1.1	0.4	0.5	0.6
Maximum	61472.8	61473.5	61477.6	61471.7	61473.5	61477.6
95% CI on Mean (Lower)	0	0	0	0	14.5	0

	Non-Zero WQ			All WQ		
	NN AE	Whitt AE	MST AE	NN AE	Whitt AE	MST AE
95% CI on Mean (Upper)	528.4	550.8	529.2	102.4	129.0	103.0

Table 5.18: p-value Comparison Matrix

	Non-Zero WQ			All WQ		
	NN AE	Whitt AE	MST AE	NN AE	Whitt AE	MST AE
NN AE		0.039	0.000		0.001	0.000
Whitt AE			0.955			0.998

Table 5.18 agrees with the results for ARE. As in *Table 5.16*, the p-values for the neural network are lower for the "All WQ" method than for the "Non-Zero WQ" method, while the p-value for Whitt is higher. Thus again, the neural network does marginally better, while Whitt's approximation does marginally worse. This, in addition to the burden of predicting whether WQ will be zero at the time of application, makes training and testing the neural network on all cases for the G/G/m (correlated arrivals) appear to be the best method.

6 Conclusions and Recommendations for Future Work

We have presented an approximation that provides some improvement over Whitt's approximation in terms of maximum absolute relative error for the independent arrivals cases. For the correlated arrivals, our approximation definitely beat Whitt's approximation. In terms of absolute relative error, our approximation failed to perform better than MST, but in terms of absolute error, our approximation performed quite well. Thus, clearly the applicability of the current approximation depends in part upon the error metrics that are of greatest importance to the practitioner. The fact that our approximation beat MST in absolute error suggests it may be possible also to perform well in terms of absolute relative error, as there is a relationship between the two. The challenge remains to determine why the neural network is experiencing large errors in absolute relative error but not in absolute error.

We obtained our results by using the standard out-of-the-box settings for the neural network software. The software has a full range of features that we will explore for future research, and that we feel optimistic will provide further improvements in our results for both independent and correlated arrivals. Besides the additional software features, we would like to explore a number of other areas to improve further our neural network single-station queueing approximation.

As we demonstrated in *Section 5.5.3*, for independent arrivals we are better off excluding cases where there is no queueing from both training and testing. As we have pointed out, to use the neural network selectively on a case-by-case basis like this requires an algorithm by which we can predict at the time of application whether WQ will be zero. In the future, we plan to compare the two methods we identified: using a cut-off level based upon Whitt's approximation for WQ and using a classifier neural network to predict whether WQ is zero. For the cutoff method, we plan to perform experimentation to determine good alpha and beta levels and compare these results with those obtained from the classifier neural network. Equipped with these results, we plan to re-address the issue of whether to exclude non-queueing cases from training and application of the neural network.

For correlated arrivals, our results reported that the mean service time was a far more significant part of the expected waiting time in the system than we anticipated. When correlation is introduced to the arrival process, we would expect queueing to increase as discussed in Patuwo (1993) and Livny (1993). We suspect initialization bias could be the culprit and the positive autocorrelation we introduced could be causing the simulation to take much longer to reach steady state. We plan to investigate this further by increasing the warm-up and replication entity counts until the expected time in system stops responding to these increases in entity counts. We hope that running the simulation longer will reduce the accuracy of MST as an estimator of WS, which would show that our neural network is a good G/G/m approximation.

As indicated in the results, we have clearly shown the feasibility of substituting a neural network approximation for part of a simulation, like a queueing station, for example. The original scope of this work involved the investigation of substituting these approximations into a larger queueing network simulation representing a logistical network. Based upon our current results, we can conclude there is hope that the objective can ultimately be achieved.

If a better single-station model can be achieved, especially for the correlated arrivals case, then current queueing network approximation methods can be married with our approach. A possible approach would be as follows. Given the network topology, demand characterization, and resource configuration, the method:

1. Randomly generates a snapshot of demand at an instant in time
2. Simulates the routing and resource-scheduling allocation and loading assignment for each demand generated
3. Loads or rejects each demand based upon the routing and scheduling analysis
4. Updates the state of the network prior to evaluating each demand
5. Exhausts the list of potential demands in the snapshot
6. Models the performance of state-of-the-network at the network, delivery and resource levels
7. Summarizes the performance measures over multiple snapshots
8. Reports statistics from the modeled network

From the demand inputs and distributions, an analytical model of the node offered load probability distribution can be derived. The node offered load distribution is sampled via Monte-Carlo simulation to generate sample observations of demand activity for the logistics network at arbitrary points in time. For each potential demand, the demand is characterized according to its distribution type and a list of demands is developed. The network is then loaded by randomly sampling from the demand list, mimicking the routing and scheduling assignments of the logistics network. These assignments require information concerning the network state, which is updated given the loading or rejecting of an individual demand from the list. After the demand list is loaded, the final state of the network is passed to a parametric decomposition-based queueing network analysis procedure (see, for example, Whitt [1983]), which uses moment-based approximations for the individual nodes within the network to estimate node performance. The node performance is then aggregated to achieve logistics network-level performance measures.

Variations of this approach have been applied to communication network design with great success. In fact, the approach can predict results within 7% of accuracy compared with actual measurements and requires computational run times of only tens of minutes, even for very large networks (see Sage and Sykes [1994]). Important issues to investigate when applying this approach to military logistical networks include:

- Improved analytical modeling of offered load distributions
- Verification and validation of demand distribution characteristics\
- Development of appropriate loading algorithms
- Examination of renewal demand arrival distributions in parametric decomposition
- Improved queueing and inventory model approximations for specific military supply situations
- Measuring and estimating the readiness of a network
- Developing and evaluating approximations to handle surge demand and/or transient conditions

The entire approach must be tested and compared to the performance of pure simulation-based approaches in order to measure the accuracy of the methods. These ideas remain as future enhancements to our current results.

7 References

1. Albin, S. L., "Approximating a Point Process by a Renewal Process, II: Superposition of Arrival Processes to Queues," *Operations Research*, No. 32, pp. 1133-1162, 1984.

2. Anderson, J. A. and E. Rosenfeld (ed.), *Neurocomputing: Foundations of Research*, Cambridge, Massachusetts: The MIT Press. 1988.

3. Cidon, I., R. Guerin, A. Khamisy, and M. Sidi, "Analysis of a Correlated Queue in Communication Systems," *IEEE Transactions on Information Theory*, Vol. 39, No. 2, pp. 456-465, March 1993.

4. Cybenko, G., "Approximation by Superpositions of a Sigmoidal Function," *Mathematical Control, Signals, and Systems*, Vol. 2, pp. 303-314, 1989.

5. Fendick, K. W., V. R. Saksena, and W. Whitt, "Dependence in Packet Queues: A Multi-Class Batch-Poisson Model," *IEEE Transactions on Communications*, Vol. 37, No. 11, pp. 1173-1183, Nov. 1989.

6. Friedman, L. W., *The Simulation Metamodel*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996.

7. Funahashi, K. I., "On the Approximate Realization of Continuous Mapping by Neural Networks," *Neural Networks*, Vol. 2, pp. 183-192, 1989.

8. Funahashi, K. I. and Y. Nakamura, "Approximation of Dynamical Systems by Continuous Time Recurrent Networks," *Neural Networks*, Vol. 6, No. 6, pp. 801-6, 1993.

9. Hecht-Nielson, R., *Neurocomputing*, Addison-Wesley Publishing Co., 1990.

10. Hornik, K., M. Stinchcombe, and H. White, "Multi-layer Feed-forward Networks Are Universal Approximators," *Neural Networks,* Vol. 2, pp. 359-366, 1989.

11. Irie, B., and S. Miyake, "Capabilities of Three-layered Perceptrons," *IEEE International Conference on Neural Networks*, Vol. 1, pp. 641-648, 1988.

12. Kimura, T., "Approximations for Multi-server Queues: System Interpolations," *Queueing Systems: Theory and Applications*, Vol. 17, Nos. 3 & 4, pp. 347-382, 1994.

13. Kleinrock, L., *Queueing Systems Volume II: Computer Applications,* Wiley Interscience, New York, pg. 49, 1976.

14. Lemke, F., "Knowledge Extraction from Data Using Self-Organizing Modeling Technologies," eSEAM'97 Conference, MacSci Tech Organization, 1997.

15. Liew, S. C., "Performance of Various Input-buffered and Output-buffered ATM Switch Design Principles under Bursty Traffic: Simulation Study," *IEEE Transactions on Communications*, Vol. 42, Nos. 2/3/4, pp. 1371-1379, February/March/April 1994.

16. Livny, M., B. Melamed, and A. K. Tsiolis, "The Impact of Autocorrelation on Queueing Systems," *Management Science*, Vol. 39. No. 3, pp. 322-339, March 1993.

17. Melamed, B. "TES: A Class of Methods for Generating Autocorrelated Uniform Variates," *ORSA Journal on Computing*, Vol. 3, No. 4, pp. 317-329, 1991.

18. Patuwo, B. E., R. L. Disney, R. L., and D. C. McNickle, "The Effect of Correlated Arrivals on Queues," *IIE Transactions*, Vol. 25, No. 3, pp. 105-110, May 1993.

19. Sage, K. M., and E. A. Sykes, "Evaluation of Routing-Related Performance for Large-Scale Packet-Switched Networks with Distributed, Adaptive Routing Policies," *Journal of Information Decision Technologies*, Vol. 19, pp. 543-562, 1994.

20. Sarle, W. S., "Neural Networks and Statistical Models," *Proceedings of the 19th Annual SAS User Group International Conference*, Dallas, TX, pp. 1538-1549, 1994.

21. Sargent, R. G, "A Historical View of Hybrid Simulation/Analytic Models," *Proceedings of the 1994 Winter Simulation Conference*, ed. J. D. Tew, S. Manivarnnan, D. A. Sadowski, and A. F. Seila, pp. 383-386, 1994.

22. Segal, M., and W. Whitt, "A Queueing Network Analyzer for Manufacturing," *Teletraffic Science for New Cost-Effective Systems, Networks, and Services, ITC-12*, M. Bonatti (ed.), Elsevier-Science, Amsterdam, pp. 1146-1152, 1989.

23. Shanthikumar, J. G., and R. G. Sargent, "A Unifying View of Hybrid Simulation/Analytic Models and Modeling," *Operations Research*, Vol. 31, No. 6, pp. 1030-1052, 1983.

24. Shin, M., R. Sargent, and A. L. Goel, "Gaussian Radial Basis Functions for Simulation Metamodeling," *Proceedings of the 2002 Winter Simulation Conference*, eds. E. Yücesan, C. H. Chen, J. L. Snowdon, and J. M. Charnes, pp. 483-488, 2002.

25. Shin, M., and A. L. Goel, "Radial Basis Function Model Development and Analysis Using the SG Algorithm," Technical Report 98-5, Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, New York, 1998.

26. Simcoe, R. J., and T. B. Pei, "Perspectives on ATM Switch Architecture and the Influence of Traffic Pattern Assumptions on Switch Design," *ACM SIGCOMM, Computer Communication Review*, Vol. 25, No. 2, pp. 95-105, April 1995.

27. Springer, M. C., and P. A. Makens, "Queueing Models for Performance Analysis: Selection of Single-Station Models," *European Journal of Operational Research*, Vol. 58, pp. 123-145, 1991.

28. Takine, T., B. Sengupta, and T. Hasegawa, "An Analysis of a Discrete-Time Queue for Broadband ISDN with Priorities among Traffic Classes," *IEEE Transactions on Communications*, Vol. 42, Nos. 2/3/4, pp. 1837-1845, Feb/Mar/Apr 1994.

29. Tin, P., "A Queueing System with Markov-Dependent Arrivals," *Journal of Applied Probability*, 22, pp. 668-677, 1985.

30. Whitt, W., "Performance of the Queueing Network Analyzer," *Bell System Technical Journal*, Vol. 62, No. 9, pp. 2817-2843, 1983.

31. Whitt, W., "The Queueing Network Analyzer," *Bell System Technical Journal*, Vol. 62, No. 9, pp. 2779-2815, 1983.

32. Whitt, W., "Planning Queueing Simulations," *Management Science*, Vol. 35, No. 11, pp. 1341-1366, Nov. 1989

33. Whitt, W., "Towards Better Multi-Class Parametric-Decomposition Approximations for Open Queueing Networks." *Annals of Operations Research*, Vol. 48, pp. 221-248, 1994.

34. Whitt, W., "Approximations for the GI/G/m Queue," *Production and Operations Management*, Vol. 2, No. 2, pp. 114-161, Spring 1993.